macromedia®

FLASH™ 5

ActionScript-Referenzhandbuch





Danksagung

Projektmanagement: Erick Vera

Texterstellung: Jody Bleyle, Mary Burger, Louis Dobrozensky, Stephanie Gowin, Marcelle Taylor und Judy Walthers Von Alten

Lektorat: Peter Fenczik, Rosana Francescato, Ann Szabla

Multimedia: George Brown, John „Zippy“ Lehnus und Noah Zilberberg

Gestaltung der Druckunterlagen und der Hilfe: Noah Zilberberg

Produktion: Christopher Basmajian, Rebecca Godbois und Noah Zilberberg

Projektmanagement Lokalisierung: Yuko Yagi

Lokalisierung: Masayo „Noppe“ Noda and Bowne Global Solutions

Besonderen Dank an: Jeremy Clark, Brian Dister und das gesamte Flash Entwicklungsteam, Michael Dominguez, Margaret Dumas, Sherri Harte, Yoshika Hedberg, Tim Hussey, Kipling Inscore, Alyn Kelley, Birgit Rathsmann, Pete Santangeli, Denise Seymour und das gesamte Flash QA Team, Cyn Taylor, Eric Wittman und Sascha Wolter

Erste Auflage: September 2000

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103, USA

INHALTSVERZEICHNIS

EINFÜHRUNG
Erste Schritte 17

Was ist neu in Flash 5 ActionScript 17
- Unterschiede zwischen ActionScript und JavaScript 18
- Textbearbeitung 18
- Punkt-Syntax 19
- Datentypen 19
- Lokale Variablen 19
- Benutzerdefinierte Funktionen 19
- Vordefinierte Objekte 19
- Aktionen für Filmsequenzen 20
- Neue Aktionen 20
- Smart-Filmsequenzen 20
- Debugger 20
- XML-Unterstützung 20
- Optimierung 20

Verwenden der Flash Hilfe für Aktionen 21

KAPITEL 1
Verwenden von ActionScript 23

Erläuterungen zum Erstellen von Skripts in ActionScript 24
- Erläuterungen zum Planen und Debuggen von Skripts 25
- Erläuterungen zu objektorientierten Skripts 26
- Erläuterungen zu den Objekten einer Filmsequenz 27
- Ablauf eines Skripts 28
- Kontrolle des Ablaufs von ActionScript 30

Terminologie in ActionScript 31

Analyse eines Beispielskripts 35

Verwenden des Bedienfeldes „Aktionen“ 38
Normaler Modus 38
Expertenmodus 40
Wechseln zwischen Bearbeitungsmodi 41
Verwenden eines externen Editors 42
Auswählen von Optionen des Bedienfeldes „Aktionen“ 42
Hervorheben und Überprüfen der Syntax 44
Erläuterungen zur Fehlerhervorhebung .45
Zuweisen von Aktionen zu Objekten 45
Zuweisen von Aktionen zu Bildern 48

KAPITEL 2

Schreiben von Skripts mit ActionScript. 51

Verwenden der ActionScript-Syntax 51
Erläuterungen zu Datentypen 56
Erläuterungen zu Variablen 59
Verwenden von Operatoren zum Ändern von Werten in Ausdrücken . .65
Schreiben von Aktionen in ActionScript 72
Steuerung des Kontrollflusses in Skripts 74
Verwenden von vordefinierten Funktionen .77
Erstellen von benutzerdefinierten Funktionen 79
Verwenden von vordefinierten Objekten 82
Verwenden von benutzerdefinierten Objekten. 86
Öffnen von Flash 4 Dateien 88
Erzeugen von Flash 4-Inhalten mit Flash 5 .90

KAPITEL 3

Erzeugen von Interaktion mit ActionScript 93

Erstellen eines benutzerdefinierten Mauszeigers. 94
Abfragen der Mausposition 96
Abfangen von Tastendrücken 97
Erstellen eines Textfeldes mit Bildlauf 99
Setzen von Farbwerten 102
Erstellen von Sound-Steuerelementen 104
Erkennen von Kollisionen. 108

KAPITEL 4

Arbeiten mit Filmsequenzen 111

Erläuterungen zur Verwendung mehrerer Zeitleisten. 112
Erläuterungen zu den hierarchischen Beziehungen zwischen Zeitleisten 114
Senden von Meldungen zwischen Zeitleisten 116
Erläuterungen zu absoluten und relativen Zielpfaden. 119
Angeben von Zielpfaden 123
Steuern von Zeitleisten durch Aktionen und Methoden 126
Erläuterungen zu Methoden und Aktionen 127
Verwenden mehrerer Methoden oder Aktionen zum Festlegen einer Zeitleiste als Ziel 128
Zuordnen einer Aktion oder Methode 128
Laden und Entladen zusätzlicher Filme .129
Ändern von Position und Darstellung einer Filmsequenz 130
Ziehen von Filmsequenzen. 131
Duplizieren und Entfernen von Filmen .132
Anhängen von Filmsequenzen 132
Erstellen von Smart-Filmsequenzen 133
Definieren von Sequenzparametern 134

KAPITEL 5

Integrieren von Flash in Webanwendungen 141

Senden und Laden von Variablen an bzw. aus einer Remote-Datei . . .141
Verwenden von loadVariables, getURL und loadMovie 145
Erläuterungen zu XML 146
Verwenden des XML-Objekts 147
Verwenden des XMLSocket-Objekts 152
Erstellen von Formularen 153
Erstellen eines Suchformulars. 154
Verwenden von Variablen in Formularen 155
Überprüfen der eingegebenen Daten 156
Senden und Empfangen von Meldungen im Flash Player 157
Verwenden von fscommand 157
Erläuterungen zu Flash Player-Methoden. 160

KAPITEL 6

ActionScript - Fehlerbehandlung 161

Richtlinien für Erstellung und Fehlerbehandlung 161
Verwenden des Debuggers. 163
Aktivieren von Debugging in einem Film. 164
Erläuterungen zur Statusleiste 165
Erläuterungen zur Anzeigeliste 166
Anzeigen und Ändern von Variablen 166
Verwenden der Überwachungsliste. 167
Anzeigen der Filmeigenschaften und Ändern bearbeitbarer Eigenschaften. 169
Verwenden des Ausgabefensters 170
Verwenden des Befehls „Objekte auflisten“ 171
Verwenden des Befehls „Variablen auflisten“ 171
Verwenden des Befehls „trace“ 172

KAPITEL 7

Referenz zu ActionScript 173

Beispieleintrag für die meisten ActionScript-Elemente 174
Beispieleintrag für Objekte 175
Inhalt der Referenz 176
— (Dekrement) 189
++ (Inkrement) 189
! (logisches NICHT) 191
!= (nicht gleich). 191
% (Modulo) 192
%= (Modulo-Zuweisung) 193
& (Bitweises UND) 193
&& (kurzgeschlossenes UND) 194
&= (bitweise UND-Zuweisung) 194
() (Klammer). 195
– (Minus) 196
* (Multiplikation) 197
*= (Multiplikationszuweisung) 197
, (Komma) 198
. (Punkt-Operator) 198
?: (bedingt) 200
/ (Division) 200
// (Kommentartrennzeichen). 201

/* (Kommentartrennzeichen) 201
/= (Divisionszuweisung) 202
[] (Arrayzugriffsoperator) 202
^ (Bitweises XODER) 203
^= (Bitweise XODER-Zuweisung) 204
{} (Objektinitialisierung) 204
| (Bitweises ODER) 206
|| (ODER) 206
|= (Bitweise ODER-Zuweisung) 207
~ (Bitweises NICHT) 208
+ (Addition) 208
+= (Additionszuweisung) 209
< (kleiner als) 210
<< (Bitweises Shift links) 211
<<= (Bitweises Shift links und Zuweisung) 211
<= (kleiner oder gleich) 212
<> (nicht gleich) 213
= (Zuweisung) 213
-= (Negationszuweisung) 214
== (gleich) 215
> (größer als) 215
>= (größer oder gleich) 216
>> (Bitweises Shift rechts) 217
>>= (Bitweises Shift rechts und Zuweisung) 218
>>> (Vorzeichenloses bitweises Shift rechts) 219
>>>= (Vorzeichenloses bitweises Shift rechts und Zuweisung) 220
add 221
_alpha 221
and 222
Array (Objekt) 222
Array.concat 224
Array.join 225
Array.length 226
Array.pop 227
Array.push 227
Array.reverse 228
Array.shift 228
Array.slice 229
Array.sort 229
Array.splice 231

Array.toString 231
Array.unshift 232
Boolean (Funktion). 232
Boolean (Objekt) 232
Boolean.toString 234
Boolean.valueOf 234
break. 234
call 235
chr 235
Color (Objekt) 236
Color.getRGB 237
Color.getTransform 237
Color.setRGB 237
Color.setTransform 238
continue 239
_currentframe 240
Date (Objekt) 241
Date.getDate. 245
Date.getDay 245
Date.getFullYear 245
Date.getHours. 246
Date.getMilliseconds. 246
Date.getMinutes 246
Date.getMonth 247
Date.getSeconds 247
Date.getTime 247
Date.getTimezoneOffset 248
Date.getUTCDate 248
Date.getUTCDay 249
Date.getUTCFullYear 249
Date.getUTCHours 249
Date.getUTCMilliseconds 249
Date.getUTCMinutes 250
Date.getUTCMonth 250
Date.getUTCSeconds 250
Date.getYear 251
Date.setDate 251
Date.setFullYear 251
Date.setHours 252
Date.setMilliseconds 252

Date.setMinutes 253
Date.setMonth 253
Date.setSeconds 253
Date.setTime. 254
Date.setUTCDate 254
Date.setUTCFullYear 254
Date.setUTCHours. 255
Date.setUTCMilliseconds 255
Date.setUTCMinutes 256
Date.setUTCMonth 256
Date.setUTCSeconds 256
Date.setYear 257
Date.toString 257
Date.UTC. 258
delete 258
do... while 260
_droptarget 261
duplicateMovieClip. 262
else { 263
eq (gleich - zeichenfolgenspezifisch) 263
escape 263
eval 264
evaluate 265
_focusrect 265
for 266
for..in 267
_framesloaded 268
fscommand 269
function 269
ge (größer oder gleich - zeichenfolgenspezifisch) 271
getProperty 271
getTimer 272
getURL. 272
getVersion 273
gotoAndPlay 274
gotoAndStop. 274
gt (größer - zeichenfolgenspezifisch) 275
_height 275
_highquality 276
if 276

ifFrameLoaded 277
#include 278
Infinity 278
int 278
isFinite 279
isNaN 279
Key (Objekt) 280
Key.BACKSPACE 281
Key.CAPSLOCK 282
Key.CONTROL 282
Key.DELETEKEY 282
Key.DOWN 283
Key.END 283
Key.ENTER 283
Key.ESCAPE 284
Key.getAscii 284
Key.getCode 284
Key.HOME 285
Key.INSERT 285
Key.isDown 285
Key.isToggled 286
Key.LEFT 286
Key.PGDN 286
Key.PGUP 287
Key.RIGHT 287
Key.SHIFT 287
Key.SPACE 288
Key.TAB 288
Key.UP 288
le (kleiner oder gleich - zeichenfolgenspezifisch) 289
length 289
_level 290
loadMovie 291
loadVariables 292
lt (kleiner als - zeichenfolgenspezifisch) 293
Math (Objekt) 294
Math.abs 296
Math.acos 296
Math.asin 297
Math.atan 297

Math.atan2 297
Math.ceil. 298
Math.cos 298
Math.E 299
Math.exp. 299
Math.floor. 300
Math.log 300
Math.LOG2E 300
Math.LOG10E. 301
Math.LN2. 301
Math.LN10. 302
Math.max 302
Math.min 302
Math.PI 303
Math.pow 303
Math.random 304
Math.round 304
Math.sin 304
Math.sqrt 305
Math.SQRT1_2 305
Math.SQRT2 306
Math.tan 306
maxscroll. 306
mbchr 307
mblength 307
mbord 308
mbsubstring 308
Mouse (Objekt). 308
Mouse.hide 309
Mouse.show 309
MovieClip (Objekt) 310
MovieClip.attachMovie. 311
MovieClip.duplicateMovieClip 312
MovieClip.getBounds 313
MovieClip.getBytesLoaded 313
MovieClip.getBytesTotal 314
MovieClip.getURL 314
MovieClip.globalToLocal 315
MovieClip.gotoAndPlay 315
MovieClip.gotoAndStop 316

MovieClip.hitTest 316
MovieClip.loadMovie 317
MovieClip.loadVariables 318
MovieClip.localToGlobal 319
MovieClip.nextFrame 319
MovieClip.play 320
MovieClip.prevFrame 320
MovieClip.removeMovieClip 320
MovieClip.startDrag 321
MovieClip.stop 321
MovieClip.stopDrag 321
MovieClip.swapDepths 322
MovieClip.unloadMovie 323
_name 323
NaN 323
ne (ungleich – zeichenfolgenspezifisch) 324
new 324
newline 325
nextFrame 325
nextScene 326
not 326
null 327
Number (Funktion) 327
Number (Objekt) 328
Number.MAX_VALUE 330
Number.MIN_VALUE 330
Number.NaN 330
Number.NEGATIVE_INFINITY 331
Number.POSITIVE_INFINITY 331
Number.toString 332
Number.valueOf 332
Object (Objekt) 332
Object.toString 333
Object.valueOf 334
onClipEvent 334
on(mouseEvent) 336
or 337
ord 338
_parent 338
parseFloat 339

parseInt 339
play 340
prevFrame 341
prevScene 341
print 342
printAsBitmap 343
_quality 344
random 345
removeMovieClip 346
return 346
_root 347
_rotation 348
scroll 348
Selection (Objekt) 349
Selection.getBeginIndex 349
Selection.getCaretIndex 350
Selection.getEndIndex 350
Selection.getFocus 350
Selection.setFocus 351
Selection.setSelection 351
set 351
setProperty 353
Sound (Objekt) 353
Sound.attachSound 355
Sound.getPan 355
Sound.getTransform 356
Sound.getVolume 356
Sound.setPan 356
Sound.setTransform 357
Sound.setVolume 360
Sound.start 361
Sound.stop 361
_soundbuftime 362
startDrag 362
stop 363
stopAllSounds 363
stopDrag 364
String (Funktion) 364
" " (Zeichenfolgentrennzeichen) 365
String (Objekt) 366

String.charAt . . . 368
String.charCodeAt . . . 368
String.concat . . . 369
String.fromCharCode . . . 369
String.indexOf . . . 369
String.lastIndexOf . . . 370
String.length . . . 370
String.slice . . . 371
String.split . . . 371
String.substr . . . 372
String.substring . . . 372
String.toLowerCase . . . 373
String.toUpperCase . . . 373
substring . . . 373
_target . . . 374
targetPath . . . 374
tellTarget . . . 375
this . . . 376
toggleHighQuality . . . 377
_totalframes . . . 377
trace . . . 378
typeof . . . 379
unescape . . . 379
unloadMovie . . . 380
updateAfterEvent . . . 380
_url . . . 381
var . . . 381
_visible . . . 382
void . . . 382
while . . . 383
_width . . . 384
with . . . 385
_x . . . 387
XML (Objekt) . . . 388
XML.appendChild . . . 391
XML.attributes . . . 392
XML.childNodes . . . 392
XML.cloneNode . . . 393
XML.createElement . . . 393
XML.createTextNode . . . 393

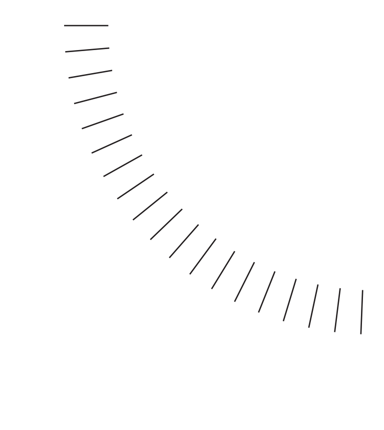
XML.docTypeDecl 394
XML.firstChild 395
XML.haschildNodes 395
XML.insertBefore 396
XML.lastChild 396
XML.load 397
XML.loaded 397
XML.nextSibling 398
XML.nodeName 398
XML.nodeType. 399
XML.nodeValue 399
XML.onLoad 400
XML.parentNode 401
XML.parseXML 401
XML.previousSibling 401
XML.removeNode 402
XML.send 402
XML.sendAndLoad. 403
XML.status 403
XML.toString 404
XML.xmlDecl. 405
XMLSocket (Objekt) 405
XMLSocket.close 407
XMLSocket.connect 408
XMLSocket.onClose 409
XMLSocket.onConnect 410
XMLSocket.onXML 411
XMLSocket.send. 412
_xmouse 413
_xscale. 413
_y 414
_ymouse 414
_yscale. 415

ANHANG A
Vorrang und Assoziativität von Operatoren 417
Operatorenliste 417

ANHANG B
Tastatureingaben und Tastencodewerte 421
Buchstaben A bis Z und Ziffern 0 bis 9 422
Tasten auf dem numerischen Ziffernblock 424
Funktionstasten 424
Andere Tasten 426

ANHANG C
Fehlermeldungen 429

INDEX 433



EINFÜHRUNG
Erste Schritte

Bei ActionScript handelt es sich um die Skriptsprache von Flash. Mit Hilfe von ActionScript können Sie Objekte in Flash Filmen steuern, Navigationselemente und interaktive Elemente erstellen und Flash erweitern, damit hoch interaktive Filme und Webanwendungen erstellt werden können.

Was ist neu in Flash 5 ActionScript

Flash 5 ActionScript bietet neue attraktive Funktionen, mit denen Sie spannende interaktive Websites mit anspruchsvollen Spielen, Formularen und Umfragen erstellen können, die Interaktivität in Echtzeit wie bei Chat-Systemen bieten.

Flash 5 ActionScript verfügt über viele neue Funktionen und Syntaxkonventionen, durch die es dem Kern der Programmiersprache JavaScript ähnelt. In diesem Handbuch werden die grundlegenden Konzepte für die Programmierung wie Funktionen, Variablen, Anweisungen, Operatoren, Bedingungen und Schleifen erläutert. In Kapitel 7 dieses Handbuchs, „Referenz zu ActionScript“, werden alle ActionScript-Elemente in detaillierten Einträgen beschrieben.

Dieses Handbuch stellt kein allgemeines Lehrbuch für die Programmierung dar. Informationen über allgemeine Konzepte bei der Programmierung und die Sprache JavaScript finden Sie in einer Vielzahl von anderen Quellen.

Die Vereinigung europäischer Computerhersteller (ECMA) hat auf der Grundlage von JavaScript ein Dokument mit dem Titel ECMA-262 herausgegeben, das als internationaler Standard für JavaScript dient. ActionScript basiert auf der in ECMA-262 enthaltenen Spezifikation, die unter http://www.ecma.ch erhältlich ist.

Dokumente und Artikel, die für das Verständnis von ActionScript nützlich sind, finden Sie in „JavaScript Developer Central“ (http://developer.netscape.com/tech/javascript/index.html) auf der Website „Netscape DevEdge Online“. Die wertvollste Ressource ist der „Core JavaScript Guide“ unter http://developer.netscape.com/docs/manuals/js/core/jsguide/index.htm.

Unterschiede zwischen ActionScript und JavaScript

Sie brauchen JavaScript nicht zu kennen, um ActionScript erlernen und verwenden zu können. Wenn Sie jedoch mit JavaScript vertraut sind, wird Ihnen ActionScript bekannt vorkommen. Einige Unterschiede zwischen ActionScript und JavaScript werden im Folgenden aufgeführt:

- ActionScript unterstützt keine browserspezifischen Objekte wie „Document“, „Window“ und „Anchor“.
- Nicht alle vordefinierten JavaScript-Objekte werden von ActionScript vollständig unterstützt.
- ActionScript unterstützt syntaktische Konstrukte, die in JavaScript nicht zulässig sind (beispielsweise die Aktionen `tellTarget` und `ifFrameLoaded` sowie Schrägstrich-Syntax).
- Einige Syntaxkonstrukte von JavaScript werden durch ActionScript nicht unterstützt, wie beispielsweise `switch`, `continue`, `try`, `catch`, `throw` und `statement`.
- ActionScript unterstützt den `Function`-Konstruktor von JavaScript nicht.
- In ActionScript kann die Aktion `eval` nur Variablenverweise ausführen.
- `toString` von `undefined` ist in JavaScript `undefined`. Aus Kompatibilitätsgründen mit Flash 4 entspricht `toString` von `undefined` bei Flash 5 „ “.
- In JavaScript ergibt eine Auswertung von `undefined` in einem numerischen Kontext den Wert `NaN`. Aus Kompatibilitätsgründen mit Flash 4 ergibt eine Auswertung von `undefined` in Flash 5 den Wert `0`.
- In ActionScript wird Unicode nicht unterstützt. Unterstützt werden die Zeichensätze ISO-8859-1 und Shift-JIS.

Textbearbeitung

Im Expertenmodus können Sie Skripts direkt im Bedienfeld „Aktionen“ eingeben. Genau wie in Flash 4, können Sie außerdem Elemente aus einem Popup-Menü oder einer Werkzeugliste auswählen.

Punkt-Syntax

Mit Punkt-Syntax können Sie die Methoden und Eigenschaften von Objekten (einschließlich Filmsequenzinstanzen und Variablen) abfragen und festlegen. Sie können anstelle der in Flash 4 verwendeten Schrägstrich-Syntax die Punkt-Syntax verwenden. Die Schrägstrich-Syntax wird nicht mehr empfohlen, aber weiterhin vom Flash Player unterstützt.

Datentypen

Flash 5 ActionScript unterstützt die folgenden Datentypen: Zeichenfolge, Zahl, Boolean, Objekt und Filmsequenz. Mit Hilfe der verschiedenen Datentypen können Sie in ActionScript mit verschiedenen Arten von Informationen arbeiten. Sie können z. B. Arrays und assoziative Arrays erstellen.

Lokale Variablen

Sie können lokale Variablen deklarieren, die nach dem Ende der Aktionsliste oder des Funktionsaufrufs ungültig werden. Dadurch können Sie sparsam mit dem Speicherplatz umgehen und Variablennamen wiederverwenden. In Flash 4 sind alle Variablen dauerhaft; sogar temporäre Variablen wie Schleifenzähler bleiben bis zum Ende des Filmes im Film enthalten.

Benutzerdefinierte Funktionen

Sie können Funktionen mit Parametern definieren, die Werte zurückgeben. Dadurch können Sie in den Skripts enthaltene Codeblöcke wiederverwenden. In Flash 4 kann Code mit Hilfe der Aktion `call` wiederverwendet werden. Es können aber keine Parameter übergeben oder Werte zurückgegeben werden.

Vordefinierte Objekte

Sie können Funktionen mit Parametern definieren, die Werte zurückgeben. Im Folgenden werden einige der vordefinierten Objekte aufgeführt:

- Das Math-Objekt enthält einen vollständigen Satz von integrierten mathematischen Konstanten und Funktionen wie `E` (Eulersche Konstante), `cos` (Kosinus) und `atan` (Arkustangens).
- Mit dem Date-Objekt können Sie auf jedem System, auf dem der Flash Player ausgeführt wird, auf das Datum und die Uhrzeit zugreifen.
- Mit dem Sound-Objekt können Sie Filmen Sounds hinzufügen und die Sounds bei der Wiedergabe eines Filmes steuern. Sie können z. B. die Lautstärke (`setVolume`) und die Balance (`setPan`) anpassen.
- Mit dem Mouse-Objekt können Sie den Standard-Mauszeiger ausblenden, so dass ein benutzerdefinierter Mauszeiger verwendet werden kann.

- Mit dem MovieClip-Objekt können Sie Filmsequenzen steuern, ohne eine Wrapperaktion wie z. B. `tellTarget` zu verwenden. Methoden wie `play`, `loadMovie` und `duplicateMovieClip` können mit Hilfe der Punkt-Syntax über den Instanznamen aufgerufen werden (z. B. `myMovieClip.play()`).

Aktionen für Filmsequenzen

Mit der Aktion `onClipEvent` können Sie den Filmsequenzinstanzen auf der Bühne direkt Aktionen zuordnen. Die Aktion „onClipEvent" enthält Ereignisse wie `load`, `Bild eingeben`, `Maus ziehen` und `Daten`, mit denen Sie neuartige Formen der Interaktivität herstellen können.

Neue Aktionen

Mit Hilfe von neuen Aktionen wie `do..while` und `for` können Sie komplexe Schleifen erstellen. Andere neue Aktionen wurden als Methoden des MovieClip-Objektes implementiert, z. B. `getBounds`, `attachMovie`, `hitTest`, `swapDepths` und `globalToLocal`.

Smart-Filmsequenzen

Smart-Filmsequenzen beinhalten Skripts, die ohne das Bedienfeld „Aktionen" geändert werden können. Über Sequenzparameter, die in der Bibliothek definiert werden, können Werte an die Smart-Filmsequenz übergeben werden.

Debugger

Mit dem Debugger können Sie Variablen- und Eigenschaftswerte anzeigen lassen, während ein Film im Filmtestmodus, im eigenständigen Flash Player oder in einem Webbrowser abgespielt wird. Auf diese Weise können Sie rasch Probleme in Ihrem ActionScript-Code finden.

XML-Unterstützung

Mit dem vordefinierten XML-Objekt können Sie ActionScript in XML-Dokumente umwandeln und diese an Server-Anwendungen übergeben. Das XML-Objekt kann auch dazu verwendet werden, XML-Dokumente in einen Flash Film zu laden und sie zu interpretieren. Mit Hilfe des XMLSocket-Objektes können Sie ständige Verbindungen mit Servern herstellen, über die XML-Daten für Echtzeitanwendungen übertragen werden können.

Optimierung

ActionScript-Code wird in Flash 5 optimiert, um die Leistung zu verbessern und Dateigrößen zu reduzieren. Aufgrund dieser Optimierungen ergibt sich bei Flash 5 oft ein kleinerer ActionScript-Bytecode als bei Flash 4.

Verwenden der Flash Hilfe für Aktionen

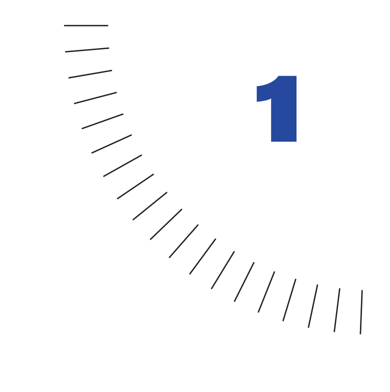
Flash 5 enthält zu jeder im Bedienfeld „Aktionen“ verfügbaren Aktion eine kontextsensitive Hilfe. Beim Verfassen von Skripts können Sie zu jeder verwendeten Aktion Informationen aufrufen.

So rufen Sie Hilfe zu Aktionen auf:

1 Wählen Sie im Bedienfeld „Aktionen“ in der Werkzeugliste eine Aktion aus.

2 Klicken Sie oben im Bedienfeld auf die Schaltfläche „Hilfe“.

Das Hilfethema für diese Aktion wird im Browser angezeigt.



KAPITEL 1
Verwenden von ActionScript

Mit ActionScript, der Skriptsprache von Flash, werden Filme interaktiv. Sie können Filme so einrichten, dass durch benutzerdefinierte Ereignisse, wie beispielsweise das Klicken auf Schaltflächen und das Drücken von Tasten, Skripts zur Auswahl der gewünschten Aktion gestartet werden. Sie können z. B. ein Skript schreiben, das – in Abhängigkeit von der Schaltfläche, auf die der Benutzer klickt – verschiedene Filme in den Flash Player lädt.

ActionScript ist einem Werkzeug vergleichbar, mit dem Sie Filme nach Ihren Wünschen erstellen können. Sie müssen nicht mit allen Einzelheiten dieses Werkzeugs vertraut sein, um ein Skript zu schreiben. Wenn Sie genau wissen, was Sie erreichen möchten, können Sie beginnen, Skripts mit einfachen Aktionen zu erstellen. Sie können dann nach und nach neue Sprachelemente einsetzen und so auch kompliziertere Aufgaben lösen.

In diesem Kapitel erhalten Sie eine Einleitung zu ActionScript als einer objektorientierten Skriptsprache sowie eine Übersicht der Begriffe von ActionScript. Außerdem finden Sie hier eine genaue Erläuterung eines Beispielskripts, anhand dessen Sie einen Überblick über ActionScript gewinnen können.

In diesem Kapitel erhalten Sie auch eine Einleitung zum Bedienfeld „Aktionen“, mit dem Sie Skripts durch Auswählen von ActionScript-Elementen erstellen können.

Erläuterungen zum Erstellen von Skripts in ActionScript

Sie können ohne eingehende Vorkenntnisse mit dem Erstellen einfacher Skripts in ActionScript beginnen. Sie müssen nur wissen, was Sie mit dem Skript erreichen möchten. Alles weitere ergibt sich aus der Auswahl geeigneter Aktionen. Am besten überzeugen Sie sich von der Einfachheit von ActionScript, indem Sie ein Skript erstellen. Mit den folgenden Schritten weisen Sie ein Skript einer Schaltfläche zu, mit der Sie die Sichtbarkeit einer Filmsequenz ändern können.

So ändern Sie die Sichtbarkeit einer Filmsequenz:

1. Wählen Sie „Fenster“ > „Allgemeine Bibliotheken“ > „Schaltflächen“ und dann „Fenster“ > „Allgemeine Bibliotheken“ > „Filmsequenzen“. Fügen Sie der Bühne eine Schaltfläche und eine Filmsequenz hinzu.
2. Wählen Sie die Filmsequenzinstanz auf der Bühne aus, und wählen Sie „Fenster“ > „Bedienfelder“ > „Instanzeigenschaften“.
3. Geben Sie im Feld „Name“ **testMC** ein.
4. Wählen Sie die Schaltfläche auf der Bühne aus, und wählen Sie „Fenster“ > „Aktionen“, um das Bedienfeld „Aktionen“ zu öffnen.
5. Klicken Sie im Bedienfeld „Objektaktionen“ auf die Kategorie „Aktionen“, um diese zu öffnen.
6. Doppelklicken Sie auf die Aktion `setProperty`, um diese in die Aktionsliste aufzunehmen.
7. Wählen Sie im Popup-Menü „Eigenschaft“ `_visible` (Sichtbarkeit) aus.
8. Geben Sie für den Parameter „Ziel“ **testMC** ein.
9. Geben Sie für den Parameter „Wert“ **0** ein.

   Dabei entsteht folgender Code:

   ```
   on (release) {
       setProperty ("testMC", _visible, false);
   }
   ```

10. Wählen Sie „Steuerung“ > „Film testen“, und klicken Sie auf die Schaltfläche, um die Filmsequenz auszublenden.

ActionScript ist eine objektorientierte Skriptsprache, d. h., Objekte werden bei einem bestimmten Ereignis durch Aktionen gesteuert. In diesem Skript ist das Ereignis das Loslassen der Maustaste, das Objekt ist die Filmsequenzinstanz `MC`, und die Aktion ist `setProperty`. Wenn der Benutzer auf die Bildschirmschaltfläche klickt, wird durch das Ereignis `release` ein Skript ausgelöst, das die Eigenschaft `_visible` des Objektes `MC` auf `false` setzt und das Objekt ausblendet.

Sie können das Bedienfeld „Aktionen“ zum Erstellen einfacher Skripts verwenden. Damit Sie alle Leistungsmerkmale von ActionScript nutzen können, ist es wichtig, die Funktionsweise dieser Sprache zu verstehen, d. h. die von dieser Sprache verwendeten Konzepte, Elemente und Regeln zum Strukturieren von Informationen und Erstellen interaktiver Filme.

In diesem Abschnitt werden der Arbeitsablauf in ActionScript, die grundlegenden Konzepte einer objektorientierten Skriptsprache, Flash Objekte und das Arbeiten mit Skripts erklärt. Außerdem wird erläutert, wo sich Skripts in einem Flash Film befinden.

Erläuterungen zum Planen und Debuggen von Skripts

Das Erstellen von Skripts für einen ganzen Film kann mit einer großen Anzahl und Vielfalt von Skripts verbunden sein. Die Auswahl der Aktionen, effektiver Skriptstrukturen und geeigneter Skriptpositionen erfordert sorgfältiges Planen und Testen, insbesondere bei zunehmender Komplexität des Filmes.

Setzen Sie sich vor dem Erstellen der Skripts ein genaues Ziel. Dies ist genauso wichtig (und im Allgemeinen genauso zeitaufwendig) wie das Entwickeln von Storyboards für Ihre Arbeit. Beginnen Sie mit dem Erstellen der gewünschten Filmhandlung wie im folgenden Beispiel:

- Ich möchte eine ganze Site mit Flash erstellen.
- Die Besucher der Site werden zur Eingabe ihres Namens aufgefordert, der dann bei verschiedenen Mitteilungen in der Site verwendet wird.
- Die Site verfügt über eine frei verschiebbare Navigationsleiste mit Schaltflächen, die mit jedem Bereich der Site verbunden sind.
- Beim Klicken auf eine Schaltfläche wird der neue Bereich im Mittelpunkt der Bühne eingeblendet.
- Eine Szene enthält ein Kontaktformular, in dem bereits der Benutzername eingetragen ist.

Wenn Sie Ihre Ziele definiert haben, können Sie die gewünschten Objekte einrichten und die Skripts zur Steuerung dieser Objekte erstellen.

Es kann eine Weile dauern, bis Skripts in der gewünschten Weise funktionieren. Oft ist dafür mehr als ein Zyklus mit Erstellen, Testen und Debuggen erforderlich. Der beste Ansatz besteht darin, mit einfachen Funktionen anzufangen und das Skript häufig zu testen. Wenn ein Teil Ihres Skripts funktioniert, wählen Sie „Speichern unter“, um eine Version der Datei (z. B. myMovie01.fla) zu speichern, und beginnen mit dem Erstellen des nächsten Teils. Mit Hilfe dieses Ansatzes können Sie Fehler schnell finden und sicherstellen, dass Ihr ActionScript auch bei komplexeren Skripts stabil bleibt.

Erläuterungen zu objektorientierten Skripts

In objektorienten Skripts werden Informationen in Gruppen, den so genannten *Klassen*, strukturiert angeordnet. Sie können mehrere Instanzen einer Klasse, so genannte *Objekte*, erzeugen und sie in Ihren Skripts verwenden. Sie können die vordefinierten Klassen von ActionScript verwenden und eigene Klassen erstellen.

Beim Erstellen einer Klasse definieren Sie alle *Eigenschaften* (die Merkmale) und *Methoden* (das Verhalten) für alle von der Klasse erzeugten Objekte genau so, wie Objekte in der Realität definiert werden. Eine Person verfügt z. B. über Eigenschaften wie Geschlecht, Größe und Haarfarbe und Methoden wie Sprechen, Gehen und Werfen. In diesem Beispiel ist „Person" eine Klasse und jede Einzelperson ein Objekt oder eine *Instanz* dieser Klasse.

Objekte in ActionScript können Daten enthalten oder auf der Bühne grafisch als Filmsequenzen dargestellt werden. Alle Filmsequenzen sind Instanzen der vordefinierten Klasse MovieClip. Jede Filmsequenzinstanz enthält alle Eigenschaften (z. B. `_height`, `_rotation`, `_totalframes`) und alle Methoden (z. B. `gotoAndPlay`, `loadMovie`, `startDrag`) der Klasse MovieClip.

Zum Definieren einer Klasse erstellen Sie eine so genannte *Konstruktor-Funktion*. Für vordefinierte Klassen sind die Konstruktor-Funktionen bereits definiert. Wenn Sie z. B. Informationen über einen Fahrradfahrer (Biker) in Ihrem Film benötigen, können Sie eine Konstruktor-Funktion `Biker` mit den Eigenschaften `time` und `distance` und der Methode `rate` erstellen, mit der Sie die Geschwindigkeit des Fahrradfahrers ermitteln können:

```
function Biker(t, d) {
   this.time = t;
   this.distance = d;
}
function Speed() {
   return this.time / this.distance;
}
Biker.prototype.rate = Speed;
```

Anschließend können Sie Kopien, d. h. Instanzen der Klasse erstellen. Mit dem folgenden Code können Sie für das Objekt `Biker` die Instanzen `emma` und `hamish` erstellen.

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5)
```

Instanzen können auch miteinander kommunizieren. Für das Objekt `Biker` können Sie eine Methode `shove` erstellen, mit der ein Fahrradfahrer (Biker) einen anderen wegstößt. (Die Instanz `emma` kann die Methode `shove` aufrufen, falls `hamish` zu nahe kommt.) Für das Übergeben von Informationen an Methoden werden Parameter (Argumente) verwendet: Beispielsweise kann die Methode `shove` die Parameter *`who`* und *`howFar`* verwenden. In diesem Beispiel schiebt `emma` den Fahrer `hamish` um 10 Pixel weg:

```
emma.shove(hamish, 10);
```

In objektorientierten Skripts können Klassen die Eigenschaften und Methoden anderer Klassen übernehmen. Dies wird als *Vererbung* bezeichnet. Mit Hilfe von Vererbung können Sie die Eigenschaften und Methoden einer Klasse erweitern oder neu definieren. Eine Klasse, die Eigenschaften oder Methoden von einer anderen Klasse erbt, wird als *Unterklasse* bezeichnet. Eine Klasse, von der Eigenschaften und Methoden an eine andere Klasse übergeben werden, wird als *Oberklasse* bezeichnet. Eine Klasse kann sowohl Unterklasse (subclass) als auch Oberklasse (superclass) sein.

Erläuterungen zu den Objekten einer Filmsequenz

Die vordefinierten Klassen von ActionScript werden als *Objekte* bezeichnet. Mit jedem Objekt haben Sie die Möglichkeit, auf einen bestimmten Informationstyp zuzugreifen. Das Objekt Date verfügt beispielsweise über Methoden (z. B. `getFullYear`, `getMonth`), mit denen sich Informationen der Systemuhr abfragen lassen. Das Objekt Sound verfügt über Methoden (z. B. `setVolume`, `setPan`), mit denen sich der Sound zu einem Film steuern lässt. Das Objekt MovieClip verfügt über Methoden, mit denen sich Filmsequenzinstanzen (z. B. `play`, `stop` und `getURL`) steuern und Informationen über deren Eigenschaften (z. B. `_alpha`, `_framesloaded`, `_visible`) abfragen und festlegen lassen.

Filmsequenzen sind die wichtigsten Objekte eines Flash Filmes, da sie über Zeitleisten verfügen, die unabhängig voneinander ausgeführt werden. Wenn beispielsweise in der Hauptzeitleiste nur ein Bild vorhanden ist und eine Filmsequenz in diesem Bild zehn Bilder enthält, werden alle Einzelbilder der Filmsequenz abgespielt. Damit können sich Instanzen als unabhängige Objekte verhalten und miteinander kommunizieren.

Jede Filmsequenzinstanz hat einen eindeutigen Instanznamen, so dass Sie zielgerichtete Aktionen mit diesen Instanzen ausführen können. Angenommen, es stehen mehrere Instanzen auf der Bühne zur Verfügung (z. B. `leftClip` und `rightClip`), und Sie möchten jeweils nur eine abspielen. Wenn Sie eine Aktion für das Abspielen einer Instanz zuweisen möchten, verwenden Sie hierfür den Namen der jeweiligen Instanz. Im folgenden Beispiel ist der Name der Filmsequenz `leftClip`:

```
leftClip.play();
```

Mit Hilfe von Instanznamen können Sie auch Filmsequenzen duplizieren, entfernen und verschieben, während ein Film abgespielt wird. Im folgenden Beispiel wird die Instanz `cartItem` dupliziert, um einen Einkaufswagen mit der Anzahl der gekauften Artikel zu füllen:

```
onClipEvent(load) {
    do {
        duplicateMovieClip("cartItem", "cartItem" + i, i);
        i  = i + 1;
    } while (i <= numberItemsPur);
}
```

Filmsequenzen verfügen über Eigenschaften, deren Werte Sie mit ActionScript dynamisch festlegen und abfragen können. Durch das Ändern und Abfragen dieser Eigenschaften können Sie das Aussehen und die Identität einer Filmsequenz ändern und somit Interaktivität erzielen. Für das folgende Skript wird z. B. die Aktion `setProperty` verwendet, um die Transparenz (Alpha-Einstellung) der `navigationBar`-Instanz auf 10 zu setzen:

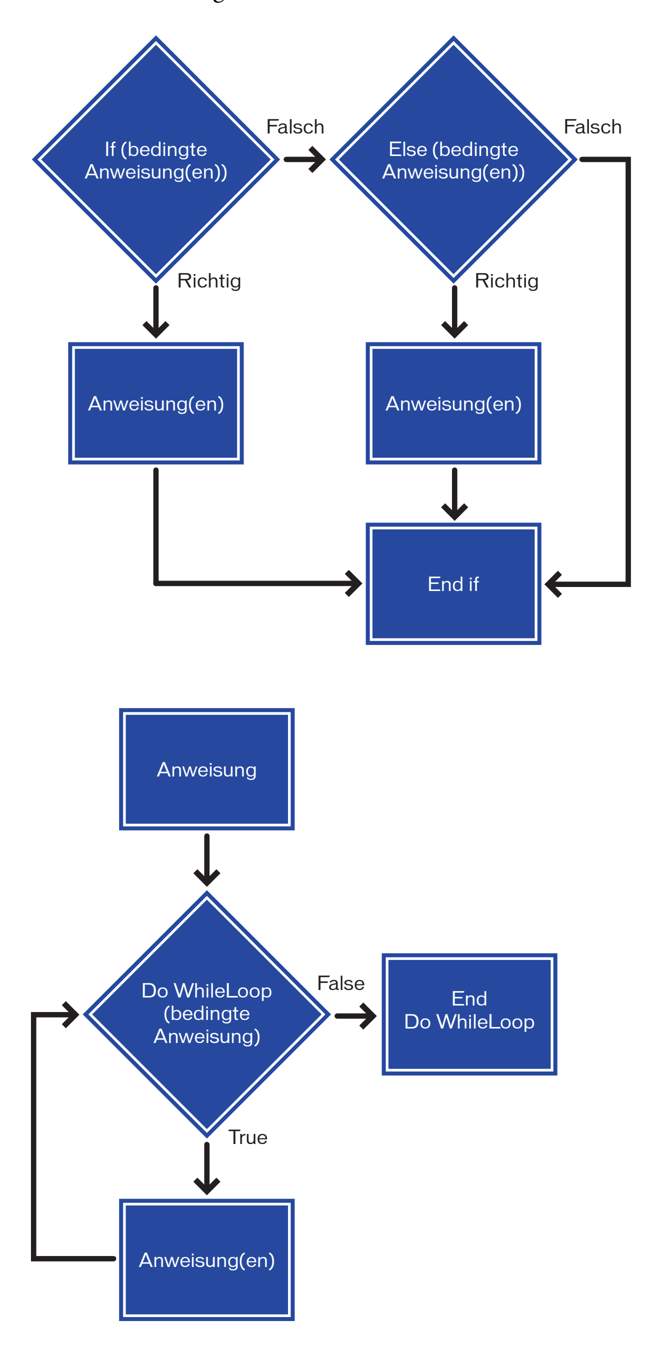
```
setProperty("navigationBar", _alpha, 10);
```

Weitere Informationen über andere Objekttypen finden Sie unter „Verwenden von vordefinierten Objekten“ auf Seite 82.

Ablauf eines Skripts

ActionScript folgt einem logischen Ablauf. ActionScript-Anweisungen werden in Flash nacheinander von der ersten bis zur letzten oder bis zu einer Anweisung ausgeführt, die ActionScript an eine andere Stelle wechseln lässt.

Einige Aktionen, mit denen ActionScript nicht zur nächsten Anweisung wechselt, sind `if`-Anweisungen, `do...while`-Schleifen und die Aktion `return`.



Eine `if`-Anweisung steuert den Ablauf eines Skripts nach der Auswertung einer bestimmten Bedingung und wird daher als bedingte Anweisung oder als „logische Verzweigung“ bezeichnet. Mit dem folgenden Code wird z. B. die Ausführung des Skripts gestartet, wenn die Variable `number` kleiner gleich 10 ist:

```
if (number <= 10) {
    alert = "Die Zahl ist kleiner gleich 10";
}
```

Sie können auch `else`-Anweisungen hinzufügen und so kompliziertere bedingte Anweisungen erstellen. Im folgenden Beispiel wird mit der `else`-Anweisung ein anderes Skript ausgeführt, wenn die Variable `number` größer als 10 ist:

```
if (number <= 10) {
    alert = "Die Zahl ist kleiner gleich 10";
} else {
    alert = "Die Zahl ist größer gleich 10";
}
```

Weitere Informationen finden Sie unter „Verwenden von if-Anweisungen“ auf Seite 75.

Mit Hilfe von Schleifen kann eine Aktion mit einer bestimmten Anzahl von Durchläufen oder bis zum Erfüllen einer bestimmten Bedingung wiederholt werden. Im folgenden Beispiel wird eine Filmsequenz fünfmal dupliziert:

```
i = 0;
do {
    duplicateMovieClip ("myMovieClip", "newMovieClip" + i, i);
    newName = eval("newMovieClip" + i);
   setProperty(newName, _x, getProperty("myMovieClip", _x) + (i * 
5));
    i = i + 1;
} while (i <= 5);
```

Weitere Informationen finden Sie unter „Wiederholen einer Aktion“ auf Seite 75.

Kontrolle des Ablaufs von ActionScript

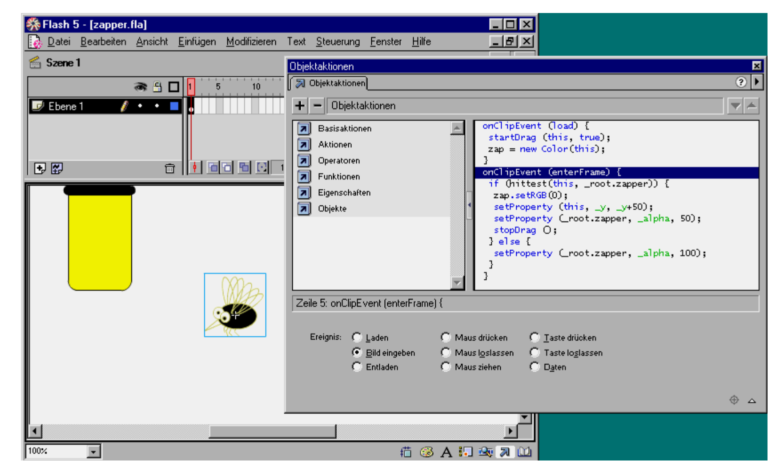
Beim Erstellen eines Skripts wird das Bedienfeld „Aktionen“ verwendet. Mit dem Bedienfeld „Aktionen“ kann das Skript einem Bild auf der Hauptzeitleiste oder auf der Zeitleiste einer beliebigen Filmsequenz zugewiesen werden. Sie können das Skript aber auch einer Schaltfläche bzw. einer Filmsequenz auf der Bühne zuweisen.

In Flash werden Aktionen je nach Zuweisung zu verschiedenen Zeiten ausgeführt:

- Die einem Bild zugewiesenen Aktionen werden ausgeführt, wenn der Abspielkopf dieses Bild erreicht.
- Die einer Schaltfläche zugewiesenen Aktionen werden in eine `on`-Handler-Aktion eingebettet.

- Die einer Filmsequenz zugewiesenen Aktionen werden in eine `onClipEvent` Handler-Aktion eingebettet.

Die Aktionen `onClipEvent` und `on` „behandeln“ oder verwalten Ereignisse und werden daher als Handler (Behandlungsroutinen) bezeichnet. (Ein Ereignis ist beispielsweise eine Mausbewegung, ein Tastendruck oder das Laden einer Filmsequenz.) Die Filmsequenz- und Schaltflächenaktionen werden ausgeführt, wenn das durch den Handler angegebene Ereignis eintritt. Sie können einem Objekt mehr als einen Handler zuordnen, wenn beim Eintreten verschiedener Ereignisse Aktionen auszuführen sind. Weitere Informationen finden Sie in Kapitel 3 unter „Erzeugen von Interaktion mit ActionScript“.



Verschiedene `onClipEvent`-Handler, die einer Filmsequenz auf der Bühne zugewiesen wurden.

Terminologie in ActionScript

Wie in jeder Skriptsprache, wird in ActionScript eine bestimmte Terminologie nach bestimmten Syntaxregeln verwendet. Mit der folgenden Liste erhalten Sie eine Einführung zu wichtigen ActionScript-Begriffen in alphabetischer Reihenfolge. Weitere Einzelheiten zu diesen Begriffen und der zugrunde liegenden Syntax finden Sie in Kapitel 2 unter „Schreiben von Skripts mit ActionScript“.

Aktionen sind Anweisungen, die einen bestimmten Vorgang bei einem abzuspielenden Film auslösen. Beispielsweise wird mit `gotoAndStop` der Abspielkopf an ein bestimmtes Bild oder an eine bestimmte Bezeichnung gesetzt. In diesem Handbuch sind die Begriffe *Aktion* und *Anweisung* austauschbar.

Argumente sind Platzhalter, mit denen Sie Werte an Funktionen übergeben können (siehe „Übergeben von Argumenten an eine Funktion“ auf Seite 80). Beispielsweise werden in der folgenden Funktion namens `welcome` zwei Werte verwendet, welche die Funktion über die Argumente `firstName` und `hobby` erhält:

```
function welcome(firstName, hobby) {
   welcomeText = "Hallo " + firstName + ", Sie mögen " + hobby;
}
```

Klassen sind Datentypen, die Sie zum Definieren eines neuen Objekttyps erstellen können. Sie definieren eine Objektklasse, indem Sie eine Konstruktor-Funktion erstellen.

Konstanten sind Elemente, die sich nicht ändern. Beispielsweise hat die Konstante `TAB` stets die gleiche Bedeutung. Mit Konstanten lassen sich Werte vergleichen.

Konstruktoren sind Funktionen, mit denen Sie die Eigenschaften und Methoden einer Klasse definieren können. Beispielsweise wird mit dem folgenden Code durch das Erstellen einer Konstruktor-Funktion namens `Circle` eine neue Klasse Circle (Kreis) erzeugt:

```
function Circle(x, y, radius){
   this.x = x;
   this.y = y;
   this.radius = radius;
}
```

Datentypen stellen einen Satz von Werten und die mit ihnen durchführbaren Operationen dar. Zeichenfolge, Zahl, die Booleschen Werte `true` und `false`, Objekt und Filmsequenz sind Datentypen von ActionScript. Weitere Informationen zu diesen Sprachelementen finden Sie unter „Erläuterungen zu Datentypen“ auf Seite 56.

Ereignissesind Aktionen, die beim Abspielen eines Filmes auftreten. So werden z. B. beim Laden einer Filmsequenz, beim Erreichen eines Bildes durch den Abspielkopf, beim Klicken auf eine Schaltfläche bzw. Filmsequenz oder bei Tastatureingaben verschiedene Ereignisse erzeugt.

Ausdrücke sind beliebige Teile einer Anweisung, die einen Wert ergeben. So stellt beispielsweise `2 + 2` einen Ausdruck dar.

Funktionen sind Blöcke von wieder verwendbarem Code, die übergebene Argumente (Parameter) verarbeiten und einen Wert zurückgeben können. Beispielsweise werden an die Funktion `getProperty` der Name einer Eigenschaft und der Instanzname einer Filmsequenz übergeben. Die Funktion gibt den Wert der Eigenschaft zurück. Die Funktion `getVersion` gibt die Version von Flash Player zurück, die zurzeit für die Filmwiedergabe verwendet wird.

Handler sind spezielle Aktionen, mit denen Ereignisse wie z. B. `mouseDown` oder `load` „behandelt“ oder verwaltet werden. Beispielsweise sind `on (onMouseEvent)` und `onClipEvent` ActionScript-Handler.

Bezeichner sind Namen, mit denen auf Variablen, Eigenschaften, Objekte, Funktionen oder Methoden verwiesen wird. Das erste Zeichen muss ein Buchstabe, Unterstrich (_) oder Dollarzeichen ($) sein. Jedes weitere Zeichen muss ein Buchstabe, eine Zahl, ein Unterstrich (_) oder ein Dollarzeichen ($) sein. Beispielsweise ist `firstName` der Name einer Variablen.

Instanzen sind Objekte, die zu einer bestimmten Klasse gehören. Jede Instanz einer Klasse enthält alle Eigenschaften und Methoden dieser Klasse. Alle Filmsequenzen sind Instanzen der Klasse MovieClip mit den Eigenschaften (z.B. `_alpha` und `_visible`) und Methoden (z.B. `gotoAndPlay` und `getURL`) dieser Klasse.

Instanznamen sind eindeutige Namen, mit denen Sie Filmsequenzinstanzen in Skripts als Ziel auswählen können. Ein Hauptsymbol in der Bibliothek könnte z.B. `counter` heißen, und die zwei Instanzen dieses Symbols im Film könnten die Instanznamen `scorePlayer1` und `scorePlayer2` haben. Durch den folgenden Code wird die Variable `score` mit Hilfe von Instanznamen in alle Filmsequenzinstanzen gesetzt:

```
_root.scorePlayer1.score += 1
_root.scorePlayer2.score -= 1
```

Schlüsselwörter sind reservierte Wörter mit einer besonderen Bedeutung. Beispielsweise ist `var` ein Schlüsselwort zum Deklarieren von lokalen Variablen.

Methoden sind die einem Objekt zugeordneten Funktionen. Nach dem Zuweisen der Funktion kann diese als Methode des zugehörigen Objektes aufgerufen werden. Im folgenden Code wird z.B. `clear` zu einer Methode des Objektes `controller`:

```
function Reset(){
   x_pos = 0;
   x_pos = 0;
}
controller.clear = Reset;
controller.clear();
```

Objekte sind eine Zusammenfassung von Eigenschaften. Jedes Objekt verfügt über einen eigenen Namen und Wert. Mit Objekten haben Sie die Möglichkeit, auf einen bestimmten Informationstyp zuzugreifen. Mit dem vordefinierten Objekt Date können beispielsweise Informationen der Systemuhr abgefragt werden.

Operatoren berechnen aus einem oder mehreren Werten einen neuen Wert. So werden z.B. mit dem Additionsoperator (+) zwei oder mehr Werte addiert, um einen neuen Wert zu erzeugen.

Zielpfade sind hierarchische Adressen von Filmsequenzinstanznamen, Variablen und Objekten in einem Film. Mit Hilfe des Bedienfeldes „Instanz“ können Sie eine Filmsequenzinstanz benennen. Die Hauptzeitleiste hat immer den Namen `_root`. Mit Hilfe eines Zielpfades können Sie eine Aktion einer bestimmten Filmsequenz zuordnen oder den Wert einer Variablen aufrufen bzw. setzen. Die folgende Anweisung ist z. B. der Zielpfad für die Variable `volume` in der Filmsequenz `stereoControl`:

```
_root.stereoControl.volume
```

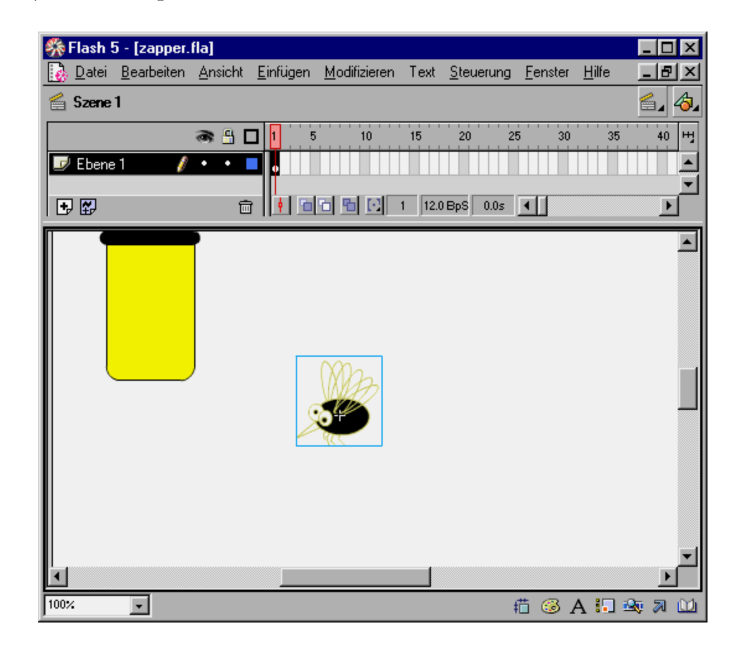
Eigenschaften sind Attribute, die ein Objekt definieren. So verfügen beispielsweise alle Filmsequenzen über die Eigenschaft `_visible`, mit der festgelegt wird, ob die jeweilige Filmsequenz sichtbar ist.

Variablen sind Bezeichner, die Werte beliebiger Datentypen enthalten können. Variablen können erstellt, geändert und aktualisiert werden. Die in ihnen gespeicherten Werte lassen sich in Skripts verwenden. Im folgenden Beispiel sind die Bezeichner auf der linken Seite des Gleichheitszeichens Variablen:

```
x = 5;
name = "Lolo";
customer.address = "Lindenweg 34";
c = new Color(mcinstanceName);
```

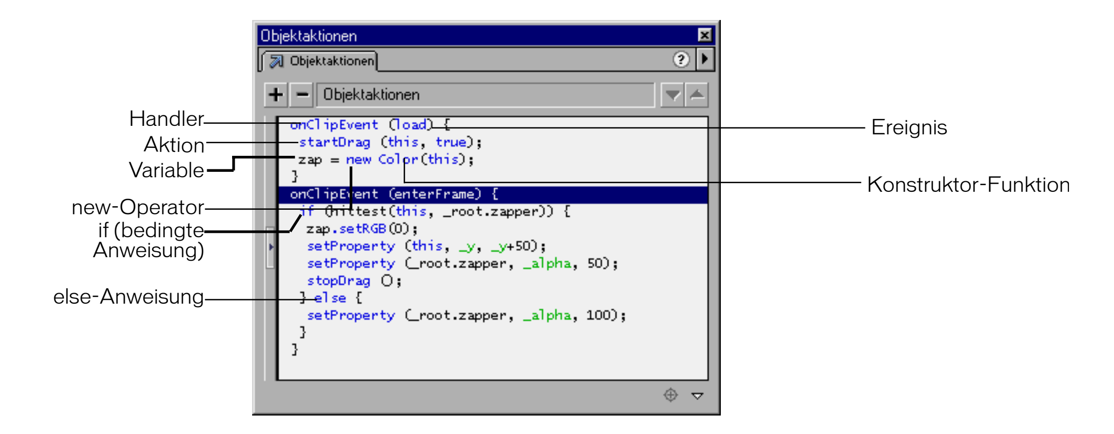
Analyse eines Beispielskripts

Wenn in diesem Beispielfilm der Benutzer den Käfer auf die Insektenfalle zieht, wechselt die Farbe des Käfers auf Schwarz, der Käfer fällt herab, und die Insektenfalle blinkt. Der Film ist ein Bild lang und enthält zwei Objekte, die Filmsequenzinstanz für den Käfer und die Filmsequenzinstanz für die Falle. Außerdem enthält jede Filmsequenz ein Bild.



Die Filmsequenzinstanzen für den Käfer und die Falle auf der Bühne in Bild 1.

Der Film enthält nur ein Skript, das wie im unten stehenden Bedienfeld „Objektaktionen“ der Instanz `bug` zugewiesen wurde:



Das Bedienfeld „Objektaktionen“ mit dem der Instanz `bug` zugewiesenen Skript.

Beide Objekte müssen Filmsequenzen sein, so dass Sie ihnen im Bedienfeld „Instanz“ Instanznamen zuweisen und sie mit ActionScript bearbeiten können. Der Instanzname des Käfers ist `bug`, und der Instanzname der Falle ist `zapper`. Im Skript wird auf den Käfer mit dem Schlüsselwort `this` verwiesen, da das Skript dem Käfer zugewiesen wurde, und das reservierte Wort `this` auf das Objekt verweist, von dem der Aufruf erfolgt.

Es sind zwei `onClipEvent`-Handler mit zwei unterschiedlichen Ereignissen vorhanden: `load` und `enterFrame`. Die Aktionen in der Anweisung `onClipEvent(load)` werden nur einmal beim Laden des Filmes ausgeführt. Die Aktionen in der Anweisung `onClipEvent(enterFrame)` werden jedesmal ausgeführt, wenn der Abspielkopf ein Bild erreicht. Selbst in einem Film mit der Länge von einem Bild wird dieses Bild vom Abspielkopf wiederholt angesteuert, wobei das Skript wiederholt abläuft. Die folgenden Aktionen treten in jedem `onClipEvent` - Handler auf:

onClipEvent(load) Durch eine `startDrag`-Aktion wird die Filmsequenz mit dem Käfer frei verschiebbar. Mit dem Operator `new` und der Color-Konstruktor-Funktion `Color` wird eine Instanz des Objektes Color erstellt und der Variablen `zap` zugewiesen:

```
onClipEvent (load) {
    startDrag (this, true);
    zap = new Color(this);
}
```

onClipEvent(enterFrame) Mit einer `if`-Anweisung wird eine `hitTest`-Aktion ausgewertet, um zu prüfen, ob sich die Käfer-Instanz (`this`) und die Fallen-Instanz (`_root.zapper`) berühren. Für diese Berechnung gibt es zwei mögliche Ergebnisse, `true` (wahr) oder `false` (falsch):

```
onClipEvent (enterFrame) {
    if (hitTest(_target, _root.zapper)) {
        zap.setRGB(0);
        setProperty (_target, _y, _y+50);
        setProperty (_root.zapper, _alpha, 50);
        stopDrag ();
    } else {
        setProperty (_root.zapper, _alpha, 100);
    }
}
```

Falls die `hitTest`-Aktion `true` zurückgibt, wird mit dem durch das `load`-Ereignis erzeugten `zap`-Objekt die Farbe des Käfers auf Schwarz gesetzt. Die *y*-Eigenschaft (`_y`) des Käfers wird auf den bisherigen Wert plus `50` gesetzt, so dass der Käfer herabfällt. Die Transparenz der Falle (`_alpha`) wird auf `50` gesetzt, so dass sie dunkler wird. Mit der `stopDrag` -Aktion wird die freie Verschiebbarkeit des Käfers aufgehoben.

Falls die `hitTest`-Aktion `false` zurückgibt, wird von der Aktion nach der `else`-Anweisung der `_alpha`-Wert der Falle auf `100` gesetzt. Da der `_alpha`-Wert vom Ausgangszustand (100) auf den Zustand für „Käfer gefangen“ (50) und wieder zurück auf den Ausgangszustand wechselt, scheint die Falle zu blinken. Die `hit-Test` -Aktion gibt `false` zurück, und die `else`-Anweisungen werden nach dem Herabfallen und Fangen des Käfers ausgeführt.

Wenn Sie den Film abspielen möchten, rufen Sie die *Flash Hilfe* auf.

Verwenden des Bedienfeldes „Aktionen“

Mit dem Bedienfeld „Aktionen“ können Sie in zwei unterschiedlichen Bearbeitungsmodi Aktionen für ein Objekt oder Bild erstellen und bearbeiten. Sie können in der Werkzeugliste vordefinierte Aktionen auswählen, Aktionen mit der Maus ziehen und ablegen und mit Schaltflächen Aktionen löschen oder neu anordnen. Im normalen Modus können Sie Aktionen mit Hilfe der Felder für Parameter (Argumente) erstellen, bei denen die richtigen Argumente vorgegeben werden. Im Expertenmodus können Sie, ähnlich dem Erstellen eines Skripts mit einem Text-Editor, Aktionen in einem Textfeld direkt erstellen und bearbeiten.

So wird das Bedienfeld „Aktionen“ aufgerufen:

Wählen Sie „Fenster“ > „Aktionen“.

Mit dem Auswählen der Instanz einer Schaltfläche oder Filmsequenz wird das Bedienfeld „Aktionen“ aktiviert. Der Titel des Bedienfeldes „Aktionen“ wechselt beim Auswählen einer Schaltfläche oder Filmsequenz auf „Objektaktionen“ und beim Auswählen eines Bildes auf „Bildaktionen“.

So wählen Sie einen Bearbeitungsmodus aus:

1 Klicken Sie bei angezeigtem Bedienfeld „Aktionen“ auf den Pfeil in der oberen rechten Ecke des Bedienfeldes, um das Popup-Menü aufzurufen.

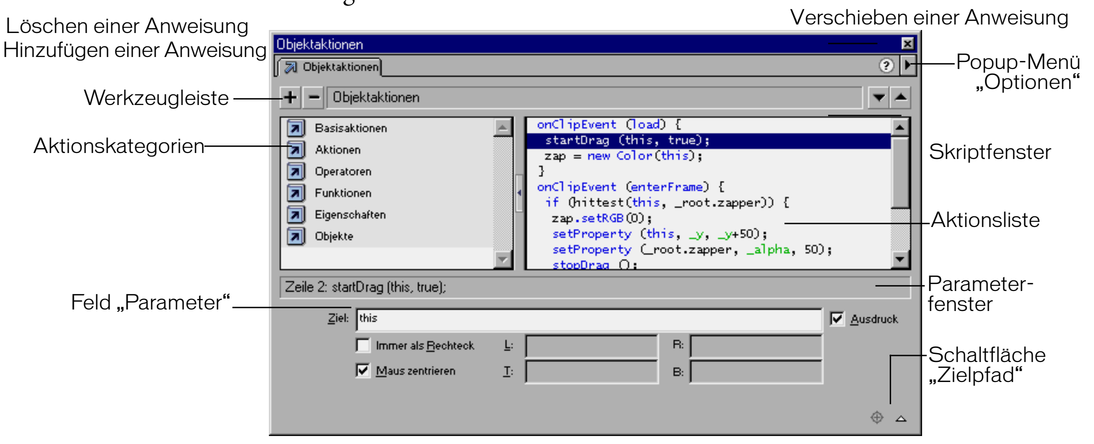
2 Wählen Sie aus dem Popup-Menü den normalen Modus oder Expertenmodus aus.

   Jedes Skript verwendet einen eigenen Modus. Sie können z. B. eine Instanz einer Schaltfläche im normalen Modus und eine andere im Expertenmodus erstellen. Wenn Sie zwischen den ausgewählten Schaltflächen wechseln, ändert sich der Modus des Bedienfeldes.

Normaler Modus

Im normalen Modus können Sie Aktionen erstellen, indem Sie auf der linken Seite des Bedienfeldes Aktionen aus der Werkzeugliste auswählen. Die Werkzeugliste enthält die Kategorien „Basisaktionen“, „Aktionen“, „Operatoren“, „Funktionen“, „Eigenschaften“ und „Objekte“. Die Kategorie „Basisaktionen“ enthält die einfachsten Flash Aktionen und ist nur im normalen Modus verfügbar. Die ausgewählten Aktionen sind auf der rechten Seite des Bedienfeldes in der Aktionsliste aufgelistet. Sie haben die Möglichkeit, Aktionen hinzuzufügen, zu löschen oder deren Reihenfolge zu ändern. Sie können außerdem in den Parameterfeldern unten im Bedienfeld Parameter (Argumente) für Aktionen eingeben.

Im normalen Modus stehen Ihnen die Steuerelemente des Bedienfeldes „Aktionen“ zur Verfügung, mit denen Sie Anweisungen in der Aktionsliste löschen oder deren Reihenfolge ändern können. Diese Steuerelemente sind besonders dann sinnvoll, wenn Sie Bild- bzw. Schaltflächenaktionen verwalten, die mehrere Anweisungen enthalten.



Das Bedienfeld „Aktionen“ im normalen Modus.

So wählen Sie eine Aktion aus:

1 Klicken Sie in der Werkzeugleiste auf eine Kategorie „Aktionen“, um die Aktionen in dieser Kategorie anzeigen zu lassen.

2 Doppelklicken Sie auf eine Aktion, oder ziehen Sie diese in das Skriptfenster.

So verwenden Sie die Parameterfelder:

1 Klicken Sie in der unteren rechten Ecke des Bedienfeldes „Aktionen“ auf die Schaltfläche „Parameter“, um die Felder anzeigen zu lassen.

2 Wählen Sie die Aktion aus, und geben Sie neue Werte in den Parameterfeldern ein, um die Parameter der vorhandenen Aktionen zu ändern.

So fügen Sie einen Zielpfad für Filmsequenzen ein:

1 Klicken Sie in der unteren rechten Ecke des Bedienfeldes „Aktionen“ auf die Schaltfläche „Zielpfad“, um das Dialogfenster „Zielpfad einfügen“ aufzurufen.

2 Wählen Sie in der Anzeigeliste eine Filmsequenz aus.

So verschieben Sie eine Anweisung in der Liste nach oben bzw. unten:

1 Wählen Sie in der Aktionsliste eine Anweisung aus.

2 Klicken Sie auf die Pfeilschaltflächen „Auf" bzw. „Ab".

So löschen Sie eine Aktion:

1 Wählen Sie in der Aktionsliste eine Anweisung aus.

2 Klicken Sie auf die Schaltfläche „Löschen (-)".

So verändern Sie die Parameter vorhandener Aktionen:

1 Wählen Sie in der Liste „Aktionen" eine Anweisung aus.

2 Geben Sie in den Parameterfeldern neue Werte ein.

Wählen Sie eines der folgenden Verfahren, um die Größe der Werkzeugliste oder der Aktionsliste zu ändern:

- Verschieben Sie den vertikalen Teiler zwischen der Werkzeugliste und der Aktionsliste.
- Doppelklicken Sie auf den Teiler, um die Werkzeugliste auszublenden. Doppelklicken Sie erneut auf den Teiler, um die Liste neu anzeigen zu lassen.
- Klicken Sie auf die Pfeilschaltflächen „Links" bzw. „Rechts" auf dem Teiler, um die Liste zu erweitern oder auszublenden.

Auch bei ausgeblendeter Werkzeugliste können Sie noch auf ihre Elemente zugreifen, indem Sie im Bedienfeld „Aktionen" links oben die Schaltfläche „Hinzufügen (+)" verwenden.

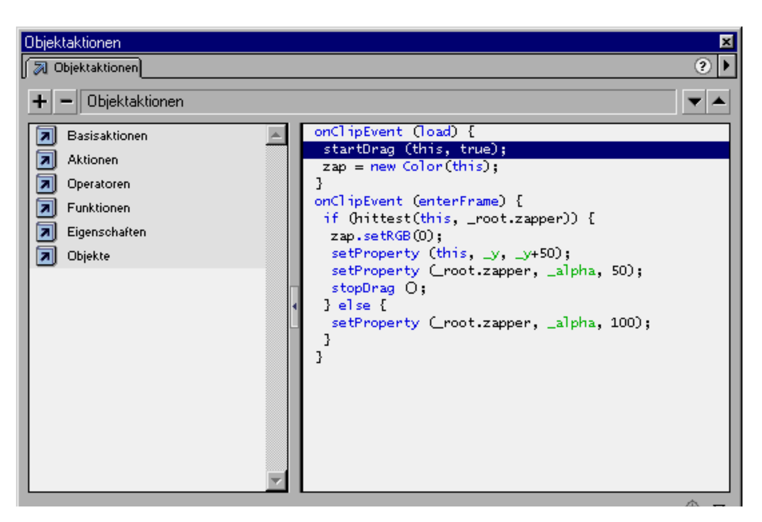
Expertenmodus

Im Expertenmodus erstellen Sie Aktionen, indem Sie auf der rechten Seite des Bedienfeldes im Textfeld ActionScript eingeben oder links in der Werkzeugliste Aktionen auswählen. Ähnlich dem Erstellen eines Skripts in einem Text-Editor, können Sie Aktionen bearbeiten, Parameter für Aktionen eingeben oder Aktionen direkt im Textfeld löschen.

Im Expertenmodus können fortgeschrittene ActionScript-Benutzer mit einem Text-Editor eigene Skripts, ähnlich wie bei JavaScript oder VBScript, bearbeiten. Der Expertenmodus unterscheidet sich vom normalen Modus in den folgenden Punkten:

- Wenn Sie im Popup-Menü „Hinzufügen" oder in der Werkzeugliste ein Element auswählen, wird das Element im Textbearbeitungsbereich eingefügt.
- Es werden keine Parameterfelder angezeigt.

- Im der Registerkarte „Aktionen“ ist nur die Schaltfläche „Hinzufügen (+)“ aktiviert.
- Die Pfeilschaltflächen „Auf“ und „Ab“ bleiben inaktiv.



Das Bedienfeld „Aktionen“ im Expertenmodus.

Wechseln zwischen Bearbeitungsmodi

Das Wechseln des Bearbeitungsmodus beim Erstellen eines Skripts kann die Formatierung des Skripts ändern. Verwenden Sie daher am besten nur einen Bearbeitungsmodus pro Skript.

Beim Wechsel vom normalen Modus zum Expertenmodus bleiben Einzüge und Formatierungen erhalten. Obwohl die im normalen Modus erstellten Skripts auch mit Fehlern in den Expertenmodus konvertiert werden können, lassen sich diese Skripts erst nach der Fehlerbeseitigung exportieren.

Der Wechsel vom Expertenmodus zum normalen Modus gestaltet sich ein wenig komplexer:

- Wenn Sie in den normalen Modus wechseln, wird das Skript von Flash neu formatiert sowie Leerzeichen und Einzüge entfernt.
- Wenn Sie in den normalen Modus und dann in den Expertenmodus zurückkehren, wird das Skript von Flash entsprechend seinem Erscheinungsbild im normalen Modus neu formatiert.
- Skripts, die im Expertenmodus erstellt wurden und Fehler enthalten, lassen sich nicht in den normalen Modus exportieren oder konvertieren. Beim Versuch, das Skript zu konvertieren, wird eine Fehlermeldung angezeigt.

So wechseln Sie zwischen Bearbeitungsmodi:

Wählen Sie in der oberen rechten Ecke des Bedienfeldes „Aktionen" im Popup-Menü den Eintrag „Normaler Modus" oder Expertenmodus" aus. Der ausgewählte Modus wird durch ein Häkchen angezeigt.

So wählen Sie eine Standardeinstellung für den Bearbeitungsmodus aus:

Wählen Sie im Menü „Bearbeiten" > „Einstellungen" aus, und wählen Sie dann im Aktionsbedienfeld „Optionen" den Eintrag „Normaler Modus" oder „Expertenmodus" aus.

Verwenden eines externen Editors

Auch wenn Sie im Expertenmodus des Bedienfeldes „Aktionen" über mehr Kontrolle beim Bearbeiten von ActionScript verfügen, haben Sie die Möglichkeit, ein Skript außerhalb von Flash zu bearbeiten. Sie können dann mit der `include`-Aktion die im externen Editor erstellten Skripts einem Skript innerhalb von Flash hinzufügen.

Mit der folgenden Anweisung wird z. B. eine Skript-Datei importiert:

```
#include "externalfile.as"
```

Die `include`-Aktion wird durch den Text der Skript-Datei ersetzt. Beim Exportieren des Filmes muss die Textdatei vorliegen.

So fügen Sie die mit dem externen Editor erstellten Skripts einem Skript in Flash hinzu:

1 Verschieben Sie `include` von der Werkzeugliste zum Skriptfenster.

2 Geben Sie im Feld „Pfad" den Pfad zur externen Datei ein.

   Der Pfad muss relativ zur FLA-Datei angegeben werden. Wenn sich z. B. „myMovie.fla" und „externalfile.as" im gleichen Ordner befinden, lautet der Pfad „externalfile.as". Falls sich „externalfile.as" in einem Unterordner namens „scripts" befindet, lautet der Pfad „scripts/externalfile.as".

Auswählen von Optionen des Bedienfeldes „Aktionen"

Mit Hilfe des Bedienfeldes „Aktionen" können Sie auf verschiedene Weise mit Skripts arbeiten. Im Skriptfenster können Sie die Schriftgröße ändern. Sie können im Bedienfeld „Aktionen" eine Textdatei mit ActionScript importieren und Aktionen als Textdatei exportieren, Text in einem Skript suchen und ersetzen sowie Syntax hervorheben, um das Lesen von Skripts und die Fehlersuche zu vereinfachen. Im Bedienfeld „Aktionen" werden Syntaxfehler und Inkompatibilitäten mit der Flash Player-Version angezeigt. Außerdem werden *veraltete* ActionScript-Elemente markiert.

Wenn nicht anders angegeben, sind diese Optionen des Bedienfeldes „Aktionen“ sowohl im normalen als auch im Expertenmodus verfügbar.

So ändern Sie im Skriptfenster die Schriftgröße:

1 Wählen Sie im Bedienfeld „Aktionen“ oben rechts im Popup-Menü den Eintrag „Schriftgröße“.

2 Wählen Sie „Klein“, „Mittel“ oder „Groß„ aus.

So importieren Sie eine Textdatei, die ActionScript enthält:

1 Wählen Sie im Bedienfeld „Aktionen“ oben rechts im Popup-Menü den Eintrag „Aus Datei importieren“.

2 Wählen Sie eine Textdatei aus, die ActionScript enthält, und klicken Sie auf „Öffnen“.

Anmerkung: Skripts mit Syntaxfehlern können nur im Expertenmodus importiert werden. Im normalen Modus wird eine Fehlermeldung angezeigt.

So exportieren Sie Aktionen als Textdatei:

1 Wählen Sie im Bedienfeld „Aktionen“ oben rechts im Popup-Menü den Eintrag „Als Datei exportieren“.

2 Wählen Sie für die Datei den gewünschten Speicherort, und klicken Sie auf „Speichern“.

So drucken Sie Aktionen:

1 Wählen Sie im Bedienfeld „Aktionen“ oben rechts im Popup-Menü den Eintrag „Drucken“.

   Das Dialogfeld „Drucken“ wird angezeigt.

2 Wählen Sie „Optionen“, und klicken Sie auf „Drucken“.

Anmerkung: Die ausgedruckte Datei enthält keine Informationen über die ursprüngliche Flash Datei. Diese Informationen sollten sinnvollerweise mit einer `comment`-Aktion in das Skript aufgenommen werden.

Wählen Sie für die Textsuche in einem Skript eine Option im Popup-Menü des Bedienfeldes „Aktionen“:

- Wechseln Sie mit „Gehe zu Zeile“ zu einer bestimmten Zeile im Skript.
- Wählen Sie „Suchen“, um Text zu suchen.
- Wählen Sie „Weitersuchen“ für die wiederholte Textsuche.

- Wählen Sie „Ersetzen“, um Text zu suchen und zu ersetzen.

  Im Expertenmodus wird mit „Ersetzen“ der gesamte Text eines Skripts durchsucht. Im normalen Modus wird mit „Ersetzen“ Text nur im Parameterfeld der jeweiligen Aktion gesucht und ersetzt. Sie können z. B. im normalen Modus nicht alle `gotoAndPlay`-Aktionen durch `gotoAndStop` ersetzen.

Anmerkung: Durchsuchen Sie die aktuelle Aktionsliste mit dem Befehl „Suchen“ oder „Ersetzen“. Verwenden Sie den Film-Explorer, um Text in allen Skripts eines Filmes zu suchen. Weitere Informationen hierzu finden Sie im *Flash Benutzerhandbuch*.

Hervorheben und Überprüfen der Syntax

Bei der Syntaxhervorhebung werden bestimmte ActionScript-Elemente farblich gekennzeichnet. Hiermit werden Syntaxfehler wie die falsche Großschreibung von Schlüsselwörtern vermieden. Falls z. B. das Schlüsselwort `typeof` als `typeOf` eingegeben wurde, wird dieses nicht blau angezeigt, so dass Sie den Fehler bemerken können. Wenn die Syntaxhervorhebung aktiviert ist, wird Text auf die folgende Weise markiert:

- Schlüsselwörter und vordefinierte Bezeichner (z. B. `gotoAndStop`, `play` und `stop`) werden blau angezeigt.
- Eigenschaften werden grün angezeigt.
- Kommentare werden magenta angezeigt.
- In Anführungszeichen gesetzte Strings werden grau angezeigt.

So schalten Sie die Syntaxhervorhebung ein und aus:

Wählen Sie im Bedienfeld „Aktionen“ oben rechts im Popup-Menü die Option „Syntax farbig anzeigen“. Die gewählte Option wird durch ein Häkchen angezeigt. Alle Skripts in einem Film werden hervorgehoben.

Es empfiehlt sich, vor dem Exportieren eines Filmes die Syntax eines Skripts auf Fehler zu überprüfen. Fehler werden in einem Fehlerbericht im Ausgabefenster angezeigt. Sie können einen Film exportieren, der Skripts mit Fehlern enthält. Sie erhalten jedoch eine Warnmeldung, dass die Skripts mit Fehlern nicht exportiert wurden.

So überprüfen Sie die Syntax eines Skripts auf Fehler:

Wählen Sie oben rechts im Bedienfeld „Aktionen“ im Popup-Menü eine Option.

- Mit „Syntax prüfen“ wird die aktuelle Aktionsliste auf Fehler überprüft.

Erläuterungen zur Fehlerhervorhebung

Im normalen Modus werden alle Syntaxfehler mit einem roten Hintergrund im Skriptfenster markiert. Dadurch lassen sich Probleme leicht finden. Wenn Sie mit dem Mauszeiger auf eine Aktion mit fehlerhafter Syntax zeigen, wird die Fehlermeldung zu dieser Aktion durch eine QuickInfo angezeigt. Beim Auswählen dieser Aktion wird die Fehlermeldung außerdem im Fenstertitel des Parameterbereichs angezeigt.

Im normalen Modus werden alle Exportinkompatibilitäten von ActionScript mit einem gelben Hintergrund im Skriptfenster markiert. Wenn z. B. die Exportversion von Flash Player auf Flash 4 gesetzt ist, wird ActionScript, das nur von Flash 5 Player unterstützt wird, gelb markiert. Die Exportversion wird im Dialogfeld „Einstellungen für Veröffentlichungen" festgelegt.

Alle veralteten Aktionen werden in der Werkzeugleiste mit grünem Hintergrund markiert. Veraltete Aktionen werden nur markiert, wenn die Exportversion von Flash auf „Flash 5" gesetzt wird.

So legen Sie die Flash Player-Exportversion fest:

1 Wählen Sie „Datei" > „Einstellungen für Veröffentlichungen".

2 Klicken Sie auf das Register „Flash".

3 Wählen Sie im Popup-Menü „Version" eine Exportversion.

Anmerkung: Die Hervorhebung von Syntaxfehlern kann nicht deaktiviert werden.

So aktivieren Sie die Hervorhebung veralteter Syntax:

Wählen Sie im Popup-Menü des Bedienfeldes „Aktionen" den Eintrag „Fehlerhafte Syntax anzeigen".

Eine vollständige Liste aller Fehlermeldungen finden Sie in Anhang D unter „Fehlermeldungen".

Zuweisen von Aktionen zu Objekten

Sie können einer Schaltfläche oder Filmsequenz eine Aktion zuweisen, die ausgeführt wird, wenn der Benutzer auf eine Schaltfläche klickt oder mit dem Mauszeiger auf sie zeigt, bzw. wenn die Filmsequenz geladen oder ein bestimmtes Bild erreicht wird. Wenn Sie die Aktion einer Instanz der Schaltfläche oder Filmsequenz zuweisen, werden andere Instanzen des Symbols nicht beeinflusst. (Informationen zum Zuweisen einer Aktion zu einem Bild finden Sie unter „Zuweisen von Aktionen zu Bildern" auf Seite 48.)

Beim Zuweisen einer Aktion zu einer Schaltfläche geben Sie die Mausereignisse an, welche die Aktion auslösen. Es können auch Tastatureingaben zum Auslösen von Aktionen zugewiesen werden. Wenn Sie einer Filmsequenz Aktionen zuweisen möchten, müssen Sie das Sequenzereignis zum Auslösen der Aktion angeben.

In den folgenden Anweisungen wird für den normalen Modus die Verwendung des Bedienfeldes "Aktionen" beim Zuweisen von Aktionen zu Objekten beschrieben.

Verwenden Sie nach dem Zuweisen den Befehl „Steuerung" > „Film testen" für die Funktionsprüfung. Die meisten Aktionen funktionieren nicht im Bearbeitungsmodus.

So weisen Sie einer Schaltfläche oder Filmsequenz eine Aktion zu:

1 Wählen Sie eine Schaltflächen- oder Filmsequenzinstanz aus. Wählen Sie anschließend „Fenster" > „Aktionen".

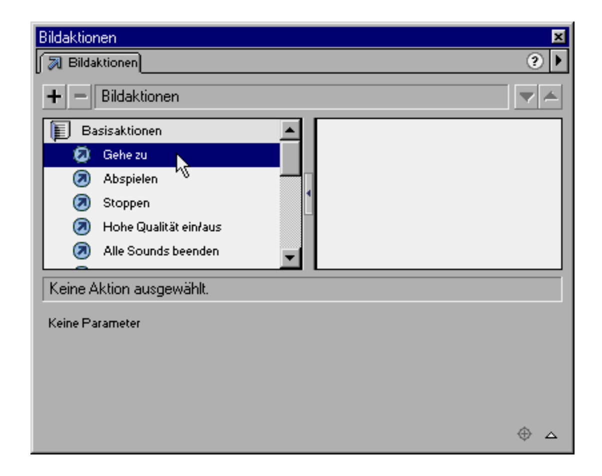
   Wenn die Auswahl weder Schaltfläche noch Filmsequenzinstanz oder Bild enthält oder wenn mehrere Objekte ausgewählt sind, wird das Bedienfeld „Aktionen" ausgeblendet.

2 Wählen Sie oben rechts im Bedienfeld „Aktionen" im Popup-Menü den normalen Modus.

3 Führen Sie zum Zuweisen einer Aktion einen der folgenden Schritte aus:

- Klicken Sie auf der linken Seite des Bedienfeldes „Aktionen" in der Werkzeugliste auf den Ordner „Aktionen". Doppelklicken Sie auf eine Aktion, um sie in die Aktionsliste auf der rechten Seite des Bedienfeldes aufzunehmen.
- Verschieben Sie eine Aktion von der Werkzeugliste in die Aktionsliste.
- Klicken Sie auf die Schaltfläche „Hinzufügen (+)", und wählen Sie im Popup-Menü eine Aktion.

- Verwenden Sie die im Popup-Menü neben jeder Aktion aufgelistete Tastenkombination.



Auswählen eines Objektes aus der Werkzeugleiste im normalen Modus

4 Wählen Sie unten im Bedienfeld in den Parameterfeldern nach Bedarf Parameter für die Aktion aus.

   Parameter ändern sich entsprechend der gewählten Aktion. Weitere Informationen über die für jede Aktion erforderlichen Parameter finden in Kapitel 7 unter „ActionScript-Verzeichnis". Klicken Sie zum Einfügen eines Zielpfades für eine Filmsequenz in ein Parameterfeld auf die Schaltfläche „Zielpfad" in der unteren rechten Ecke des Bedienfeldes „Aktionen". Weitere Informationen finden Sie in Kapitel 4 unter „Arbeiten mit Filmsequenzen".

5 Wiederholen Sie die ggf. die Schritte 3 und 4, um weitere Aktionen zuzuweisen.

So testen Sie eine Objektaktion:

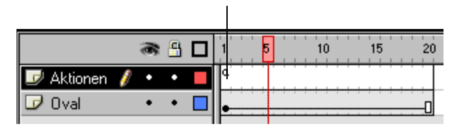
Wählen Sie „Steuerung" > „Film testen".

Zuweisen von Aktionen zu Bildern

Eine Bildaktion wird einem Schlüsselbild zugewiesen, um eine bestimmte Aktion zu veranlassen, sobald ein Film ein Schlüsselbild erreicht. Um z.B. in der Zeitleiste zwischen Bild 20 und 10 eine Schleife zu erstellen, fügen Sie Bild 20 die folgende Bildaktion hinzu:

```
gotoAndPlay (10);
```

Es empfiehlt sich, Bildaktionen in einer eigenen Ebene zu speichern. Bilder, die Aktionen enthalten, werden in der Zeitleiste mit einem kleinen *a* gekennzeichnet.



Ein „a" in einem Schlüsselbild kennzeichnet eine Bildaktion.

Verwenden Sie nach dem Zuweisen einer Aktion den Befehl „Steuerung" > „Film testen" für die Funktionsprüfung. Die meisten Aktionen funktionieren nicht im Bearbeitungsmodus.

In den folgenden Anweisungen wird für den normalen Modus die Verwendung des Bedienfeldes „Aktionen" beim Zuweisen von Bildaktionen beschrieben. (Weitere Informationen über das Zuweisen einer Aktion zu einer Schaltfläche oder Filmsequenz finden Sie unter „Zuweisen von Aktionen zu Objekten" auf Seite 45.)

So weisen Sie einem Schlüsselbild eine Aktion zu:

1 Wählen Sie in der Zeitleiste ein Schlüsselbild aus. Wählen Sie anschließend „Fenster" > „Aktionen".

   Falls das ausgewählte Bild kein Schlüsselbild ist, wird die Aktion dem vorhergehenden Schlüsselbild zugewiesen. Falls die Auswahl kein Bild ist oder mehrere Schlüsselbilder enthält, wird das Bedienfeld „Aktionen" abgeblendet.

2 Wählen Sie oben rechts im Bedienfeld „Bildaktionen" im Popup-Menü den normalen Modus.

3 Führen Sie zum Zuweisen einer Aktion einen der folgenden Schritte aus:

- Klicken Sie auf der linken Seite des Bedienfeldes „Aktionen" in der Werkzeugliste auf den Ordner „Aktionen". Doppelklicken Sie auf eine Aktion, um sie in die Aktionsliste auf der rechten Seite des Bedienfeldes aufzunehmen.
- Verschieben Sie eine Aktion von der Werkzeugliste in die Aktionsliste.

- Klicken Sie auf die Schaltfläche „Hinzufügen (+)“, und wählen Sie im Popup-Menü eine Aktion.
- Verwenden Sie die im Popup-Menü neben jeder Aktion aufgelistete Tastenkombination.
- Wählen Sie unten im Bedienfeld in den Parameterfeldern nach Bedarf Parameter für die Aktion aus.

4 Wiederholen Sie die ggf. die Schritte 3 und 4, um weitere Aktionen zuzuweisen.

So testen Sie eine Bildaktion:

Wählen Sie „Steuerung“ > „Film testen“.

KAPITEL 2
Schreiben von Skripts mit ActionScript

Beim Erstellen von Skripts mit ActionScript können Sie verschiedene Arbeitsmethoden verwenden. Bei einfachen Aktionen können Sie das Bedienfeld „Aktionen" im normalen Modus verwenden und Skripts mit Hilfe von Optionen aus Menüs und Listen zusammensetzen. Wenn Sie mit ActionScript jedoch leistungsfähigere Skripts schreiben möchten, müssen Sie die Funktionsweise von ActionScript als Programmiersprache verstehen.

ActionScript besteht, wie andere Scriptsprachen auch, aus Komponenten wie vordefinierten Objekten und Funktionen, und es gestatten Ihnen, eigene Objekte und Funktionen zu erstellen. ActionScript hat eigene Syntaxregeln, reserviert Schlüsselwörter, stellt Operatoren zur Verfügung und gestattet es Ihnen, Variablen zum Speichern und Abrufen von Informationen zu verwenden.

Syntax und Stil von ActionScript sind dem von JavaScript sehr ähnlich. Flash 5 konvertiert ActionScript aus früheren Flash Versionen.

Verwenden der ActionScript-Syntax

ActionScript besitzt Grammatik- und Zeichensetzungsregeln, die festlegen, welche Zeichen und Wörter sinngebend sind und in welcher Reihenfolge sie geschrieben werden können. Im Deutschen wird beispielsweise ein Satz durch einen Punkt beendet. In ActionScript wird eine Anweisung durch ein Semikolon abgeschlossen.

Im Folgenden finden Sie allgemeine Regeln für ActionScript. Für die meisten ActionScript-Begriffe gelten bestimmte Regeln, die Sie im entsprechenden Eintrag in Kapitel 7, „ActionScript-Verzeichnis", finden.

Punkt-Syntax

In ActionScript verweist ein Punkt (.) auf die zu einem Objekt oder einer Filmsequenz gehörigen Eigenschaften und Methoden. Er dient auch zur Angabe des Zielpfades einer Filmsequenz oder einer Variablen. Ein Punkt-Syntax-Ausdruck beginnt mit dem Namen des Objektes oder der Filmsequenz, gefolgt von einem Punkt, und endet mit der Eigenschaft, Methode oder Variablen, die Sie angeben möchten.

Beispielsweise bezeichnet die Eigenschaft `_x` einer Filmsequenz die Position der *x*-Achse der Filmsequenz auf der Bühne. Der Ausdruck `ballMC._x` bezieht sich auf die Eigenschaft `_x` der Filmsequenzinstanz `ballMC`.

Ein weiteres Beispiel: `submit` ist eine Variable, die in der Filmsequenz `form` gesetzt wird, die in der Filmsequenz `shoppingCart` verschachtelt ist. Der Ausdruck `shoppingCart.form.submit = true` setzt die Variable `submit` der Instanz `form` auf `true`.

Ein Ausdruck für eine Methode eines Objektes oder einer Filmsequenz folgt demselben Muster. Beispielsweise bewegt in der nachfolgenden Anweisung die Methode `play` der Instanz `ballMC` den Abspielkopf in der Zeitleiste von `ballMC`, wie in der folgenden Anweisung:

```
ballMC.play();
```

Die Punkt-Syntax kennt zwei spezielle Aliase: `_root` und `_parent`. Das Alias `_root` bezieht sich auf die Hauptzeitleiste. Mit dem Alias `_root` können Sie einen absoluten Zielpfad erzeugen. Beispielsweise ruft die folgende Anweisung die Funktion `buildGameBoard` in der Filmsequenz `functions` in der Hauptzeitleiste auf:

```
_root.functions.buildGameBoard();
```

Mit dem Alias `_parent` verweisen Sie auf eine Filmsequenz, in der die aktuelle Filmsequenz verschachtelt ist. Mit `_parent` können Sie einen relativen Zielpfad erzeugen. Wenn beispielsweise die Filmsequenz `dog` in der Filmsequenz `animal` verschachtelt ist, gibt die folgende von der Instanz `dog` ausgehende Anweisung an, dass `animal` gestoppt werden soll:

```
_parent.stop();
```

Siehe Kapitel 4, „Arbeiten mit Filmsequenzen“.

Die Schrägstrich-Syntax

Die Schrägstrich-Syntax diente in Flash 3 und 4 zur Angabe des Zielpfades einer Filmsequenz oder einer Variablen. Diese Syntax wird von Flash 5 Player zwar noch unterstützt, ihre Verwendung empfiehlt sich jedoch nicht. Bei der Schrägstrich-Syntax werden Schrägstriche anstelle von Punkten zur Angabe des Pfades einer Filmsequenz oder einer Variabeln verwendet. Um anzuzeigen, dass es sich um eine Variable handelt, wird einem Begriff wie im folgenden Beispiel ein Doppelpunkt vorangestellt:

```
myMovieClip/childMovieClip:myVariable
```

Sie können denselben Zielpfad auf folgende Weise mit Punkt-Syntax schreiben:

```
myMovieClip.childMovieClip.myVariable
```

Die Schrägstrich-Syntax wurde meist für die Aktion `tellTarget` verwendet, von deren Verwendung ebenfalls abgeraten wird.

Anmerkung: Die Aktion `with` wird jetzt der Aktion `tellTarget` vorgezogen, da sie besser mit der Punkt-Syntax kompatibel ist. Weitere Informationen finden Sie in den einzelnen Einträgen in Kapitel 7, „ActionScript-Verzeichnis."

Geschweifte Klammern

ActionScript-Anweisungen werden mit geschweiften Klammern (`{ }`) zu Blöcken zusammengefasst, wie im folgenden Script:

```
on(release) {
    myDate = new Date();
    currentMonth = myDate.getMonth();
}
```

Siehe „Schreiben von Aktionen in ActionScript" auf Seite 72.

Semikola

```
Eine ActionScript-Anweisung endet mit einem Semikolon. WennSie das
abschließende Semikolon jedoch vergessen, kompiliert Flash das
Skript trotzdem erfolgreich. Die folgenden Anweisungen enden
beispielsweise mit einem Semikolon:
column = passedDate.getDay();
row    = 0;
```

Dieselben Anweisungen könnten ohne abschließendes Semikolon geschrieben werden:

```
column = passedDate.getDay()
row    = 0
```

Runde Klammern

Bei der Definition einer Funktion werden die Argumente in runde Klammern eingeschlossen:

```
function myFunction (name, age, reader){
   ...
}
```

Beim Aufruf einer Funktion werden die Argumente der Funktion in runden Klammern übergeben, beispielsweise:

```
myFunction ("Steve", 10, true);
```

Sie können mit runden Klammern auch die Reihenfolge von Operationen in ActionScript beeinflussen oder ActionScript-Anweisungen leichter lesbar machen. Siehe „Vorrang des Operators“ auf Seite 66.

Runde Klammern dienen auch zum Auswerten eines Ausdrucks auf der linken Seite einer Punkt-Syntax. In der folgenden Anweisung bewirken die runden Klammern beispielsweise, dass `new color(this)` ausgewertet wird und ein neues Farb-Objekt erzeugt:

```
onClipEvent(enterFrame) {
   (new Color(this)).setRGB(0xffffff));
}
```

Ohne den Einsatz von runden Klammern müssten Sie eine Anweisung einfügen, die die Auswertung bewirkt:

```
onClipEvent(enterFrame) {
   myColor = new Color(this);
   myColor.setRGB(0xffffff);
}
```

Groß- und Kleinbuchstaben

ActionScript unterscheidet nur bei Schlüsselwörtern zwischen Groß- und Kleinschreibung. In allen anderen Fällen können Sie Groß- oder Kleinbuchstaben beliebig verwenden. Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
cat.hilite = true;
CAT.hilite = true;
```

Sie sollten sich jedoch konsistente Großschreibungskonventionen angewöhnen, beispielsweise die in diesem Buch verwendeten, um Funktions- und Variablennamen beim Lesen von ActionScript-Code leichter zu erkennen.

Wenn Sie bei Schlüsselwörtern nicht korrekt zwischen Groß- und Kleinschreibung unterscheiden, ist Ihr Skript fehlerhaft. Wenn die Option „Syntax farbig anzeigen“ im Bedienfeld „Aktionen“ eingeschaltet ist, werden richtig geschriebene Schlüsselwörter blau angezeigt. Weitere Informationen erhalten Sie unter „Schlüsselwörter“ auf Seite 55 und „Hervorheben und Überprüfen der Syntax“ auf Seite 44.

Kommentare

Im Bedienfeld „Aktionen“ können Sie einem Bild oder einer Schaltfläche mit Hilfe einer `Kommentar`-Anweisung Anmerkungen hinzufügen, um die gewünschten Ergebnisse der Aktion festzuhalten. Kommentare erleichtern auch die Weitergabe von Informationen, wenn in einem Entwicklerteam gearbeitet oder Beispiele weitergegeben werden.

Wenn Sie die Aktion `Kommentar` wählen, werden die Zeichen `//` in das Skript eingefügt. Selbst ein einfaches Skript ist leichter verständlich, wenn Sie bei der Erstellung Anmerkungen einfügen:

```
on(release) {
    // neues Date-Objekt erzeugen
    myDate = new Date();
    currentMonth = myDate.getMonth();
    // Monatszahl in Monatsnamen konvertieren
    monthName = calcMonth(currentMonth);
    year = myDate.getFullYear();
    currentDate = myDate.getDat ();
}
```

Kommentare werden im Skriptfenster in der Farbe Rosa angezeigt. Sie können beliebig lang sein, ohne dass die Größe der exportierten Datei dadurch beeinflusst wird, und sie unterliegen nicht den Regeln bezüglich ActionScript-Syntax oder Schlüsselwörtern.

Schlüsselwörter

In ActionScript sind einige Wörter für eine bestimmte Verwendung reserviert. Diese können daher nicht als Namen für Variablen, Funktionen oder Labels verwendet werden. In der folgenden Tabelle sind alle ActionScript-Schlüsselwörter aufgeführt:

break	for	new	var
continue	function	return	void
delete	if	this	while
else	in	typeof	with

Weitere Informationen zu bestimmten Schlüsselwörtern finden Sie in den entsprechenden Einträgen in Kapitel 7, „ActionScript-Verzeichnis.“

Konstanten

Eine Konstante ist eine Eigenschaft, deren Wert sich nie ändert. Konstanten sind in der Symbolleiste „Aktionen“ und in Kapitel 7, „ActionScript-Verzeichnis“, in Großbuchstaben aufgeführt.

Die Konstanten `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE` und `TAB` sind beispielsweise Eigenschaften eines Key-Objektes und verweisen auf Tasten auf der Computertastatur. Mit der folgenden Anweisung können Sie prüfen, ob der Anwender die Eingabetaste drückt:

```
if(keycode() == Key.ENTER) {
   alert = "Sind Sie bereit?"
   controlMC.gotoAndStop(5);
}
```

Erläuterungen zu Datentypen

Ein Datentyp beschreibt die Art der Informationen, die eine Variable oder ein ActionScript-Element enthalten kann. Es gibt zwei Arten von Datentypen: Grundtyp und Referenz. Die Grunddatentypen (Zeichenfolge, Zahl und Boolean) haben einen konstanten Wert und können daher die eigentlichen Werte des Elements enthalten, das sie repräsentieren. Die Referenzdatentypen (Filmsequenz und Objekt) haben veränderliche Werte und enthalten daher nur Referenzen auf die eigentlichen Werte des Elements. Variablen, die Grunddatentypen enthalten, verhalten sich in bestimmten Situationen anders als Variablen, die Referenzdatentypen enthalten. Siehe „Verwenden von Variablen in einem Skript“ auf Seite 62.

Es folgt eine Aufzählung aller Datentypen und ihrer Regeln. Zu ausführlicher besprochenen Datentypen werden Verweise angegeben.

Zeichenfolge

Eine Zeichenfolge ist eine Folge von Zeichen, beispielsweise Buchstaben, Ziffern oder Satzzeichen. Zeichenfolgen werden in ActionScript in einfache oder doppelte Anführungszeichen eingeschlossen. Zeichenfolgen werden als Zeichen und nicht als Variablen behandelt. In der folgenden Anweisung beispielsweise ist `"L7"` eine Zeichenfolge:

```
favoriteBand = "L7";
```

Mit dem Additionsoperator (`+`) können Sie zwei Zeichenfolgen *verketten* (verbinden). ActionScript behandelt Leerzeichen am Anfang oder Ende einer Zeichenfolge als tatsächlichen Bestandteil der Zeichenfolge. Der folgende Ausdruck enthält ein Leerzeichen hinter dem Komma:

```
greeting = "Willkommen," + firstName;
```

Obwohl ActionScript bei Verweisen auf Variablen, Instanznamen und Bildbezeichnungen nicht zwischen Groß- und Kleinbuchstaben unterscheidet, wird bei Literalzeichenfolgen zwischen Groß- und Kleinschreibung unterschieden. Die folgenden zwei Anweisungen legen beispielsweise unterschiedlichen Text in die angegebenen Textfeld-Variablen ab, da "Hallo" und "HALLO" Zeichenketten-Literale sind.

```
invoice.display = "Hallo";
invoice.display = "HALLO";
```

Wenn Sie ein Anführungszeichen in eine Zeichenfolge einfügen möchten, müssen Sie einen umgekehrten Schrägstrich (\) voranstellen. Dies wird als „Escape-Sequenz“ bezeichnet. Es gibt noch andere Zeichen, die in ActionScript nur durch besondere Escape-Sequenzen dargestellt werden können. Die folgende Tabelle enthält alle Escape-Zeichen von ActionScript:

Escape-Sequenz	Zeichen
`\b`	Rückschritttaste (Backspace, ASCII 8)
`\f`	Seitenvorschub (Form Feed, ASCII 12)
`\n`	Zeilenvorschub (Line Feed, ASCII 10)
`\r`	Wagenrücklauf (Carriage Return, ASCII 13)
`\t`	Tabulator (Tab, ASCII 9)
`\"`	Doppeltes Anführungszeichen
`\'`	Einfaches Anführungszeichen
`\\`	Umgekehrter Schrägstrich
`\000 - \377`	Ein oktaler Byte-Wert
`\x00 - \xFF`	Ein hexadezimaler Byte-Wert
`\u0000 - \uFFFF`	Ein 16-Bit-Unicode-Zeichen in Hexadezimal-Schreibweise

Zahl

Der Datentyp Zahl ist eine doppelt-genaue Gleitkommazahl. Sie können auf Zahlen die arithmetischen Operatoren Addition (+), Subtraktion (-), Multiplikation, (*), Division (/), Modulo (%), Inkrementieren (++) und Dekrementieren (--) anwenden. Sie können Zahlen auch mit den Methoden des vordefinierten Mathe-Objektes verarbeiten. Im folgenden Beispiel wird die Methode `sqrt` (square root, Quadratwurzel) zur Berechnung der Quadratwurzel der Zahl 100 eingesetzt:

```
Math.sqrt(100);
```

Siehe „Numerische Operatoren“ auf Seite 66.

Boolean

Die Werte des Typs Boolean sind `true` (wahr) oder `false` (falsch). ActionScript konvertiert die Werte `true` und `false` gegebenenfalls in 1 und 0. Werte vom Typ Boolean werden meist zusammen mit logischen Operatoren in ActionScript-Anweisungen verwendet, die über Vergleiche den Ablauf eines Skripts steuern. Im folgenden Skript wird beispielsweise der Film abgespielt, wenn die Variable `password` den Wert `true` hat:

```
onClipEvent(enterFrame) {
    if ((userName == true) && (password == true)){
        play();
    }
}
```

Siehe „Verwenden von if-Anweisungen“ auf Seite 75 und „Logische Operatoren“ auf Seite 68.

Objekt

Ein Objekt ist eine Sammlung von Eigenschaften. Eine Eigenschaft hat einen Namen und einen Wert. Der Wert einer Eigenschaft kann ein beliebiger Flash Datentyp sein, sogar der Datentyp Objekt. Dadurch können Sie ein Objekt in ein anderes einfügen („verschachteln“). Zum Angeben von Objekten und deren Eigenschaften wird der Punkt-Operator (.) verwendet. In dem folgenden Beispiel ist `hoursWorked` eine Eigenschaft von `weeklyStats`, das wiederum eine Eigenschaft von `employee` ist:

```
employee.weeklyStats.hoursWorked
```

Die in ActionScript vordefinierten Objekte ermöglichen den Zugriff und das Manipulieren bestimmter Arten von Informationen. Das Math-Objekt besitzt beispielsweise Methoden zur Ausführung von mathematischen Operationen mit den übergebenen Zahlen. In diesem Beispiel wird die Methode `sqrt` verwendet:

```
squareRoot = Math.sqrt(100);
```

Das ActionScript-Objekt MovieClip besitzt Methoden zur Steuerung von Instanzen von Filmsequenz-Symbolen auf der Bühne. In diesem Beispiel werden die Methoden `play` und `nextFrame` verwendet:

```
mcInstanceName.play();
mc2InstanceName.nextFrame();
```

Sie können auch eigene Objekte erzeugen, die die Informationen in Ihrem Film strukturieren. Wenn Sie einem Film mit ActionScript Interaktivität hinzufügen möchten, benötigen Sie viele verschiedene Informationen: Sie könnten beispielsweise Folgendes benötigen: einen Benutzernamen, die Geschwindigkeit eines Balles, die Namen der Artikel in einem Einkaufswagen, die Anzahl der geladenen Bilder, die Postleitzahl des Benutzers und die zuletzt gedrückte Taste. Wenn Sie Objekte selbst definieren, können Sie diese Informationen in Gruppen aufteilen, Ihre Skripterstellung vereinfachen und Ihre Skripte wiederverwenden. Weitere Informationen finden Sie unter „Verwenden von benutzerdefinierten Objekten“ auf Seite 86.

Filmsequenz

Filmsequenzen sind Symbole, die Animationen in einem Flash Film abspielen können. Dies ist der einzige Datentyp, der auf ein grafisches Element verweist. Mit dem Datentyp Filmsequenz können Sie über die Methoden des MovieClip-Objektes Filmsequenz-Symbole steuern. Methoden werden folgendermaßen mit dem Punkt-Operator aufgerufen:

```
myClip.startDrag(true);
parentClip.childClip.getURL("http://www.macromedia.com/support/"
+ product);
```

Erläuterungen zu Variablen

Eine Variable ist ein Behälter für Informationen. Der Behälter an sich bleibt dabei immer gleich, nur der Inhalt kann sich ändern. Wenn Sie beim Abspielen eines Filmes den Wert einer Variablen ändern, können Sie Informationen über die Aktionen des Benutzers aufzeichnen und speichern, die Veränderung von Werten beim Abspielen eines Filmes aufzeichnen und feststellen, ob eine bestimmte Bedingung wahr oder falsch ist.

Es empfiehlt sich, einer Variablen bei der Definition einen bestimmten Wert zuzuweisen. Diese so genannte „Initialisierung“ einer Variablen wird oft im ersten Bild eines Filmes vorgenommen. Wenn eine Variable initialisiert wird, ist es leichter, den Wert der Variablen beim Abspielen des Filmes zu verfolgen und zu vergleichen.

Variablen können jeden beliebigen Datentyp enthalten: Zahl, Zeichenfolge, Boolean, Objekt oder Filmsequenz. Der Datentyp einer Variablen bestimmt die Art, wie sich der Wert der Variablen bei einer Zuweisung innerhalb eines Skripts ändert.

Zu den am häufigsten in Variablen gespeicherten Informationen gehören URLs, Benutzernamen, Ergebnisse mathematischer Operationen, Zähler für bestimmte Ereignisse sowie die Information, ob auf eine bestimmte Schaltfläche geklickt wurde. Jeder Film bzw. jede Filmsequenz hat einen eigenen Satz von Variablen, wobei jede Variable einen eigenen Wert enthält, der von den Werten der Variablen aus anderen Filmen oder Filmsequenzen unabhängig ist.

Bezeichnen einer Variablen

Der Name einer Variablen muss folgende Bedingungen erfüllen:

- Er muss ein Bezeichner sein. (Siehe Identifiers.)
- Er darf kein Schlüsselwort und kein Literal vom Typ Boolean (`true` oder `false`) sein.
- Er muss innerhalb seines Gültigkeitsbereichs eindeutig sein. (Siehe „Festlegen des Gültigkeitsbereichs einer Variablen“ auf Seite 61.)

Festlegen des Variablentyps

In Flash müssen Sie bei der Definition einer Variablen nicht explizit festlegen, ob sie eine Zahl, eine Zeichenfolge oder einen anderen Datentyp enthalten soll. Flash bestimmt den Datentyp einer Variablen dann, wenn dieser ein Wert zugewiesen wird:

```
x = 3;
```

Im Ausdruck `x = 3` wertet Flash das Element auf der rechten Seite des Operators aus und erkennt den Typ Zahl. Bei einer späteren Zuweisung könnte sich der Typ von `x` ändern. Beispielsweise wird bei `x = "hallo"` der Typ von `x` auf Zeichenfolge geändert. Eine Variable, der kein Wert zugewiesen wurde, hat den Typ `undefined` (nicht definiert).

Wenn ein Ausdruck es erfordert, werden Datentypen in ActionScript automatisch konvertiert. Wenn Sie beispielsweise der Aktion `trace` einen Wert übergeben, konvertiert `trace` diesen Wert automatisch in eine Zeichenfolge und sendet ihn an das Ausgabefenster. In Ausdrücken mit Operatoren konvertiert ActionScript Datentypen bei Bedarf. Wenn der Operator + beispielsweise zusammen mit einer Zeichenfolge verwendet wird, wird erwartet, dass der andere Operand ebenfalls eine Zeichenfolge ist:

```
"Next in line, number” + 7
```

ActionScript konvertiert die Zahl 7 in die Zeichenfolge `"7"` und fügt sie an das Ende der ersten Zeichenfolge an:

```
"Next in line, number 7”
```

Wenn Sie Fehler in Skripts beheben (Debugging), empfiehlt es sich, den Datentyp eines Ausdrucks oder einer Variablen zu bestimmen, um ein bestimmtes Verhalten zu verstehen. Dies geschieht mit dem Operator `typeof`, beispielsweise folgendermaßen:

```
trace(typeof(variableName));
```

Zum Konvertieren einer Zeichenfolge in einen numerischen Wert wird die Funktion `Number` verwendet. Zum Konvertieren eines numerischen Wertes in eine Zeichenfolge wird die Funktion `String` verwendet. Siehe die entsprechenden Einträge in Kapitel 7, „Referenz zu ActionScript“ auf Seite 173.

Festlegen des Gültigkeitsbereichs einer Variablen

Der Gültigkeitsbereich ("Scope") einer Variablen bezieht sich auf den Bereich, in dem die Variable bekannt ist und in dem auf sie verwiesen werden kann. Variablen können in ActionScript entweder global oder lokal sein. Eine globale Variable wird von allen Zeitleisten gemeinsam verwendet, eine lokale Variable ist nur innerhalb ihres Codeblocks (zwischen den geschweiften Klammern) verfügbar.

Mit der Anweisung `var` können Sie eine lokale Variable in einem Skript deklarieren. Die Variablen `i` und `j` werden beispielsweise gern als Schleifenzähler verwendet. Im folgenden Beispiel wird `i` als lokale Variable verwendet, die nur innerhalb der Funktion `makeDays` vorhanden ist:

```
function makeDays(){
   var i
   for( i = 0; i < monthArray[month]; i++ ) {

      _root.Days.attachMovie( "DayDisplay", i, i + 2000 );

      _root.Days[i].num = i + 1;
      _root.Days[i]._x = column * _root.Days[i]._width;
      _root.Days[i]._y = row * _root.Days[i]._height;

      column = column + 1;

      if (column == 7 ) {

         column = 0;
         row = row + 1;
      }
   }
}
```

Mit lokalen Variablen vermeiden Sie auch Namenskollisionen, die in dem Film zu Fehlern führen können. Wenn Sie beispielsweise `name` als lokale Variable verwenden, könnten Sie dort in einem Zusammenhang einen Benutzernamen und in einem anderen den Namen einer Filmsequenz-Instanz speichern. Es kann keine Kollision auftreten, da sich die Variablen in verschiedenen Gültigkeitsbereichen befinden.

Es empfiehlt sich, im Hauptabschnitt einer Funktion lokale Variablen zu verwenden, damit die Funktion als unabhängiger Code arbeiten kann. Eine lokale Variable kann nur innerhalb ihres eigenen Codeblocks geändert werden. Wenn ein Ausdruck innerhalb einer Funktion eine globale Variable verwendet, könnte der Wert der Funktion extern geändert werden, was wiederum die Funktion ändern würde.

Variablendeklaration

Zur Deklaration von globalen Variablen können Sie die Aktion `setVariables` oder den Zuweisungsoperator (=) verwenden. Beide Methoden führen zum gleichen Ergebnis.

Lokale Variablen werden mit Hilfe der `var`-Anweisung innerhalb des Hauptabschnitts einer Funktion deklariert. Lokale Variablen sind im Block gültig und werden am Ende des Blocks ungültig. Lokale Variablen, die nicht innerhalb eines Blocks deklariert werden, werden am Ende des Skripts ungültig.

Anmerkung: Die Aktion `call` erzeugt ebenfalls einen neuen Gültigkeitsbereich für lokale Variablen in dem Skript, das sie aufruft. Wenn das aufgerufene Skript beendet ist, wird dieser Gültigkeitsbereich für lokale Variablen aufgehoben. Von dieser Vorgehensweise ist jedoch abzuraten, da die Aktion `call` durch die Aktion `with` ersetzt wurde, die besser mit der Punkt-Syntax kompatibel ist.

Wenn Sie den Wert einer Variablen überprüfen möchten, können Sie den Wert mit der Aktion `trace` an das Ausgabefenster senden. Beispielsweise sendet `trace(hoursWorked)` den Wert der Variablen `hoursWorked` im Filmtestmodus an das Ausgabefenster. Sie können Werte von Variablen auch im Debugger im Filmtestmodus überprüfen und setzen. Weitere Informationen finden Sie in Kapitel 6, „Troubleshooting ActionScript“.

Verwenden von Variablen in einem Skript

Eine Variable muss deklariert werden, bevor sie in einem Ausdruck verwendet werden kann. Wenn Sie wie in dem folgenden Beispiel eine undeklarierte Variable verwenden, ist der Wert der Variablen `undefined`, und Ihr Skript erzeugt eine Fehlermeldung:

```
getURL(myWebSite);
myWebSite = "http://www.shrimpmeat.net";
```

Die Anweisung, die die Variable `myWebSite` deklariert, muss an erster Stelle stehen, damit die Variable in der Aktion `getURL` durch einen Wert ersetzt werden kann.

Sie können den Wert einer Variablen in einem Skript häufig ändern. Der Datentyp, den die Variable enthält, beeinflusst das Verhalten der Variablen bei Änderungen. Grunddatentypen wie Zeichenfolgen und Zahlen werden als Wert übergeben. Das heißt, dass der eigentliche Inhalt einer Variablen an die Variable übergeben wird.

In dem folgenden Beispiel wird `x` auf 15 gesetzt, und dieser Wert wird in `y` kopiert. Wenn `x` auf 30 geändert wird, behält `y` den Wert 15, da `y` nicht in `x` nach seinem Wert sucht, sondern den Wert von `x` enthält, der übergeben wurde.

```
var x = 15;
var y = x;
var x = 30;
```

Im nächsten Beispiel enthält die Variable `in` den Wert eines Grunddatentyps, 9. Daher wird der eigentliche Wert an die Funktion `sqrt` (Quadratwurzel) übergeben, und der zurückgegebene Wert ist 3:

```
function sqrt(x){
  return x * x;
}

var in = 9;
var out = sqr(in);
```

Der Wert der Variablen `in` ändert sich nicht.

Der Datentyp Objekt kann derart viele und komplexe Informationen enthalten, dass eine Variable dieses Typs nicht den eigentlichen Wert, sondern einen Verweis auf den Wert enthält. Dieser Verweis wirkt wie ein Alias, der auf den Inhalt der Variablen zeigt. Wenn die Variable ihren Wert benötigt, fordert der Verweis den Wert an und gibt ihn zurück, ohne den Wert an die Variable zu übertragen.

Es folgt ein Beispiel zur Übergabe mit Hilfe eines Verweises:

```
var myArray = ["tom", "dick"];
var newArray = myArray;
myArray[1] = "jack";
trace(newArray);
```

Der obige Code erzeugt ein Array-Objekt namens `myArray`, das zwei Elemente enthält. Die Variable `newArray` wird erzeugt und erhält einen Verweis auf `myArray`. Wenn das zweite Element von `myArray` geändert wird, betrifft dies jede Variable, die auf dieses verweist. Die Aktion `trace` würde `["tom", "jack"]` an das Ausgabefenster senden.

Im nächsten Beispiel enthält `myArray` ein Array-Objekt. Es wird also mit Hilfe eines Verweises an die Funktion `zeroArray` übergeben. Die Funktion `zeroArray` ändert den Inhalt des Arrays in `myArray`.

```
function zeroArray (array){
  var i;
  for (i=0; i < array.length; i++) {
    array[i] = 0;
  }
}

var myArray = new Array();
myArray[0] = 1;
myArray[1] = 2;
myArray[2] = 3;

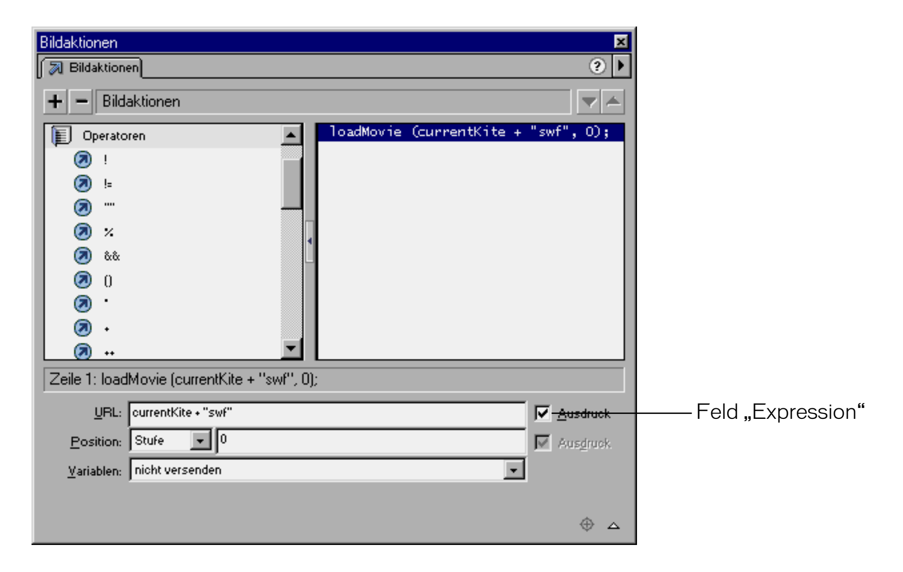
var out = zeroArray(myArray)
```

Die Funktion `zeroArray` akzeptiert ein Array-Objekt als Argument und setzt alle Elemente des Arrays auf 0. Sie kann das Array modifizieren, da das Array mit Hilfe eines Verweises übergeben wird.

Alle Verweise auf beliebige Objekte außer Filmsequenzen heißen *harte Referenzen*, da ein referenziertes Objekt nicht gelöscht werden kann. Ein Verweis auf eine Filmsequenz ist eine besondere Art von Referenz, die als *weiche Referenz* bezeichnet wird. Für eine weiche Referenz ist das referenzierte Objekt nicht unbedingt erforderlich. Wenn eine Filmsequenz mit einer Aktion wie `removeMovieClip`gelöscht wird, funktionieren Referenzen auf diese Sequenz nicht mehr.

Verwenden von Operatoren zum Ändern von Werten in Ausdrücken

Ein Ausdruck ist eine durch Flash auswertbare Anweisung, die einen Wert zurückgibt. Sie erstellen einen Ausdruck, indem Sie Operatoren und Werte verbinden oder eine Funktion aufrufen. Wenn Sie im Bedienfeld „Aktionen“ einen Ausdruck im normalen Modus schreiben, muss das Kontrollkästchen „Ausdruck“ im Bedienfeld „Parameter“ aktiviert sein. Andernfalls wird das Feld einen Zeichenfolgen-Literal enthalten.



Operatoren sind Zeichen, die festlegen, wie Werte in einem Ausdruck kombiniert, verglichen oder geändert werden. Die Elemente, auf die der Operator angewendet wird, werden als *Operanden* bezeichnet. In der folgenden Anweisung addiert der Operator + beispielsweise den Wert eines numerischen Literals zum Wert der Variablen `foo`; `foo` und `3` sind die Operanden:

```
foo + 3
```

In diesem Abschnitt werden die allgemeinen Regeln für häufig verwendete Operatortypen beschrieben. Einzelheiten zu den hier erwähnten Operatoren sowie zu besonderen Operatoren, die nicht in diese Kategorien fallen, finden Sie in Kapitel 7, „ActionScript-Verzeichnis“.

Vorrang des Operators

Wenn zwei oder mehr Operatoren in derselben Anweisung auftreten, haben manche Operatoren Vorrang vor anderen Operatoren. ActionScript folgt bei der Ausführung der Operatoren einer genau festgelegten Hierarchie. Eine Multiplikation wird z. B immer vor einer Addition ausgeführt. Elemente in Klammern haben jedoch Vorrang vor Multiplikationen. Im folgenden Beispiel führt ActionScript die Multiplikation zuerst aus, da keine Klammern vorhanden sind:

```
total = 2 + 4 * 3;
```

Das Ergebnis ist 14.

Wenn die Addition jedoch geklammert wird, führt ActionScript zuerst die Addition aus:

```
total = (2 + 4) * 3;
```

Das Ergebnis ist 18.

Eine Tabelle mit allen Operatoren und ihren Vorrangregeln finden Sie in Anhang B, „Vorrang und Assoziativität von Operatoren".

Assoziativität von Operatoren

Wenn zwei oder mehr Operatoren den gleichen Vorrang haben, bestimmt ihre Assoziativität die Reihenfolge, in der sie ausgeführt werden. Es gibt Assoziativität von links nach rechts (Linksassoziativität) und Assoziativität von rechts nach links (Rechtsassoziativität). Die Multiplikation ist beispielsweise linksassoziativ. Die folgenden Anweisungen sind also gleichwertig:

```
total = 2 * 3 * 4;
total = (2 * 3) * 4;
```

Eine Tabelle aller Operatoren und ihrer Assoziativität finden Sie in Anhang B, „Vorrang und Assoziativität von Operatoren".

Numerische Operatoren

Numerische Operatoren addieren, subtrahieren, multiplizieren, dividieren und führen weitere arithmetische Operationen durch. Klammern und Minus-Zeichen sind arithmetische Operatoren. In der folgenden Tabelle sind die numerischen Operatoren in ActionScript aufgeführt:

Operator	Ausgeführte Operation
+	Addition
*	Multiplikation
/	Division

%	Modulo
-	Subtraktion
++	Inkrement
--	Dekrement

Vergleichsoperatoren

Vergleichsoperatoren vergleichen Werte von Ausdrücken und geben einen Wert vom Typ Boolean (`true` oder `false`) zurück. Diese Operatoren werden meist in Schleifen und bedingten Anweisungen verwendet. Im folgenden Beispiel wird, wenn die Variable `score` den Wert `100`hat, ein bestimmter Film geladen, andernfalls wird ein anderer Film geladen:

```
if (score == 100){
   loadMovie("winner.swf", 5);
} else {
   loadMovie("loser.swf", 5);
   }
```

In der folgenden Tabelle sind die Vergleichsoperatoren in ActionScript aufgeführt:

Operator	Ausgeführte Operation
<	Kleiner als
>	Größer als
<=	Kleiner oder gleich
>=	Größer oder gleich

Zeichenfolgenoperatoren

Der Operator + hat eine besondere Wirkung, wenn er auf Zeichenfolgen angewendet wird: Er verkettet zwei Zeichenfolgen-Operanden. Die folgende Anweisung addiert beispielsweise `"Herzlichen Glückwunsch, "` und `"Donna!"`:

```
"Herzlichen Glückwunsch, " + "Donna!"
```

Das Ergebnis ist `"Herzlichen Glückwunsch, Donna!"` Wenn nur einer der Operanden des Operators + eine Zeichenfolge ist, konvertiert Flash den anderen Operanden in eine Zeichenfolge.

Vergleichsoperatoren, >, >=, < und <= haben ebenfalls eine besondere Wirkung, wenn sie auf Zeichenfolgen angewendet werden. Diese Operatoren vergleichen zwei Zeichenfolgen nach ihrer alphabetischen Reihenfolge. Die Vergleichsoperatoren vergleichen Zeichenfolgen jedoch nur dann, wenn beide Operanden Zeichenfolgen sind. Wenn nur einer der Operanden eine Zeichenfolge ist, konvertiert ActionScript beide Operanden in Zahlen und führt einen numerischen Vergleich durch.

Anmerkung: Die Datentypisierung von ActionScript in Flash 5 ermöglicht es, dass derselbe Operator auf unterschiedliche Datentypen angewendet werden kann. Es nicht mehr erforderlich, die Zeichenfolgen-Operatoren von Flash 4 zu verwenden (beispielsweise `eq`, `ge` und `lt`), es sei denn, Sie exportieren einen Flash 4 Film.

Logische Operatoren

Logische Operatoren vergleichen Werte vom Typ Boolean (`true` und `false`) und geben einen dritten Wert vom Typ Boolean zurück. Wenn beide Operanden z.B `true`ergeben, gibt der logische Operator UND (`&&`) das Ergebnis`true` zurück. Wenn einer oder beide Operanden `true` ergeben, gibt der logische Operator ODER (`||`) das Ergebnis `true`zurück. Logische Operatoren werden häufig in Verbindung mit Vergleichsoperatoren verwendet, um die Bedingung für eine `if`-Aktion festzulegen. Im folgenden Skript-Beispiel wird die `if`-Aktion ausgeführt, wenn beide Bedingungen wahr (true) sind:

```
if ((i > 10) && (_framesloaded > 50)){
   play()
}
```

In der folgenden Tabelle sind die logischen Operatoren in ActionScript aufgeführt:

Operator	Ausgeführte Operation
&&	Logisches UND
||	Logisches ODER
!	Logisches NICHT

Bit-Operatoren

Bit-Operatoren konvertieren intern Gleitkommazahlen in 32-Bit-Ganzzahlen, da diese leichter zu verarbeiten sind. Welche Bit-Operation genau ausgeführt wird, hängt vom Operator ab. Alle Bit-Operationen bestimmen jedoch alle Ziffern einer Gleitkommazahl getrennt, um einen neuen Wert zu berechnen.

In der folgenden Tabelle sind die Bit-Operatoren in ActionScript aufgeführt:

Operator	Ausgeführte Operation
&	Bitweises UND
|	Bitweises ODER
^	Bitweises XOR (entweder-oder)
~	Bitweises NICHT
<<	Shift links
>>	Shift rechts
>>>	Shift rechts, auffüllen mit 0

Gleichheits- und Zuweisungsoperatoren

Der Gleichheitsoperator (==) bestimmt, ob die Werte zweier Operanden übereinstimmen bzw. die Operanden identisch sind. Der Vergleich gibt einen Wert vom Typ Boolean (`true` oder `false`) zurück. Wenn die Operanden Zeichenfolgen, Zahlen oder Werte vom Typ Boolean sind, werden ihre Werte verglichen. Wenn die Operanden Objekte oder Arrays sind, werden sie mit Hilfe einer Referenz verglichen.

Der Zuweisungsoperator (=) weist einer Variablen einen Wert zu. Ein Beispiel:

password = „Sk8tEr“;

Mit dem Zuweisungsoperator können auch Zuweisungen an mehrere Variablen in demselben Ausdruck erfolgen. In der folgenden Anweisung wird den Variablen c und d der Wert b zugewiesen:

```
a = b = c = d;
```

Mit zusammengesetzten Zuweisungsoperatoren können mehrere Operationen hintereinander ausgeführt werden. Zusammengesetzte Operatoren führen eine Operation auf beiden Operanden aus und weisen dann dem ersten Operanden das Ergebnis zu. Die folgenden Anweisungen sind beispielsweise gleichwertig:

```
x += 15;
x = x + 15;
```

In der folgenden Tabelle sind die Gleichheits- und Zuweisungsoperatoren in ActionScript aufgeführt:

Operator	Ausgeführte Operation
==	Gleichheit
!=	Ungleichheit
=	Zuweisung
+=	Addition und Zuweisung
-=	Subtraktion und Zuweisung
*=	Multiplikation und Zuweisung
%=	Modulo und Zuweisung
/=	Division und Zuweisung
<<=	Bitweises Shift links und Zuweisung
>>=	Bitweises Shift rechts und Zuweisung
>>>=	Shift rechts, auffüllen mit 0 und Zuweisung
^=	Bitweises XOR und Zuweisung
|=	Bitweises ODER und Zuweisung
&=	Bitweises UND und Zuweisung

Punkt-Operator und Array-Zugriffsoperator

Der Punkt-Operator (`.`) und der Array-Zugriffsoperator (`[]`) werden zum Zugriff auf beliebige vordefinierte oder selbstdefinierte Eigenschaften eines ActionScript-Objektes verwendet, einschließlich der Eigenschaften einer Filmsequenz.

Der Punkt-Operator verwendet den Namen eines Objektes auf der linken und den Namen einer Eigenschaft oder einer Variablen auf der rechten Seite. Der Name der Eigenschaft oder der Variablen darf weder eine Zeichenfolge noch eine Variable sein, die zu einer Zeichenfolge ausgewertet wird. Er muss ein Bezeichner sein. Es folgen zwei Beispiele zur Verwendung des Punkt-Operators:

```
year.month = "Juni";
year.month.day = 9;
```

Der Punkt-Operator und der Array-Zugriffsoperator erfüllen dieselbe Aufgabe, nur erwartet der Punkt-Operator als Eigenschaft einen Bezeichner, während der Array-Zugriffsoperator seinen Inhalt als Namen auswertet und auf den Wert der so benannten Eigenschaft zugreift. Die folgenden beiden Beispiele greifen auf dieselbe Variable `velocity` in der Filmsequenz `rocket` zu:

```
rocket.velocity;
rocket["velocity"];
```

Mit dem Array-Zugriffsoperator können Namen von Instanzen und Variablen dynamisch gesetzt und abgerufen werden. Im folgenden Code-Beispiel wird der Ausdruck innerhalb der Operators `[]` ausgewertet und das Resultat der Auswertung als Name der zu ermittelnden Variablen der Filmesequenz `name` verwendet:

```
name["mc" + i ]
```

Wenn Sie mit Schrägstrich-Syntax von Flash 4 ActionScript vertraut sind, können Sie dies auch mit Hilfe der Funktion `eval` erreichen, wie in folgendem Beispiel:

```
eval("mc" & i);
```

Der Array-Zugriffsoperator kann auch auf der linken Seite einer Zuweisungsanweisung eingesetzt werden. So können Namen von Instanzen, Variablen und Objekten dynamisch gesetzt werden, wie im folgenden Beispiel:

```
name[index] = "Gary";
```

Dies entspricht wieder der Flash 4 ActionScript-Schrägstrich-Syntax:

```
Set Variable: "name:" & index = "Gary"
```

Der Array-Zugriffsoperator kann auch verschachtelt werden, um mehrdimensionale Arrays zu simulieren.

```
chessboard[row][column]
```

Dies entspricht der folgenden Schrägstrich-Syntax:

```
eval("chessboard/" & row & ":" & column)
```

Anmerkung: Wenn Sie ActionScript schreiben möchten, das mit Flash 4 Player kompatibel ist, können Sie die Aktion `eval` mit dem Operator `add` verwenden.

Schreiben von Aktionen in ActionScript

Aktionen sind die Anweisungen oder Befehle von ActionScript. Mehrere demselben Bild oder Objekt zugewiesene Aktionen erzeugen ein Skript. Aktionen können unabhängig voneinander ausgeführt werden, wie bei den folgenden Anweisungen:

```
swapDepths("mc1", "mc2");
gotoAndPlay(15);
```

Sie können Aktionen auch verschachteln, indem eine Aktion in der anderen ausgeführt wird. Dadurch können die Aktionen einander beeinflussen. Im folgenden Beispiel teilt die `if` -Aktion der Aktion `gotoAndPlay` mit, wann sie stattfinden soll:

```
if (i >= 25) {
    gotoAndPlay(10);
}
```

Aktionen können den Abspielkopf in der Zeitleiste bewegen (`gotoAndPlay`), den Ablauf eines Skripts durch Schleifen (`do while`) oder logische Bedingungen (`if`) steuern oder neue Funktionen und Variablen erzeugen (`function`, `setVariable`). In der folgenden Tabelle sind alle ActionScript-Aktionen aufgeführt:

break	evaluate	include	print	stopDrag
call	for	loadMovie	printAsBitmap	swapDepths
comment	for...in	loadVariables	removeMovieClip	tellTarget
continue	fsCommand	nextFrame nextScene	return	toggleHigh-Quality
delete	function	on	setVariable	stopDrag
do...while	getURL	onClipEvent	setProperty	trace
duplicateMovie-Clip	gotoAndPlay gotoAndStop	play	startDrag	unloadMovie
else	if	prevFrame	stop	var
else if	ifFrameLoaded	prevScene	stopAllSounds	while

Beispiele zur Syntax und Verwendung aller Aktionen finden Sie in den einzelnen Einträgen in Kapitel 7, „ActionScript-Verzeichnis“.

Anmerkung: In diesem Buch ist der ActionScript-Begriff *Aktion* (action) gleichbedeutend mit dem JavaScript-Begriff *Anweisung* (statement).

Schreiben eines Zielpfades

Um eine Aktion zur Steuerung eines Filmsequenz oder eines geladenen Filmes zu verwenden, müssen Sie Namen und Adresse, *Zielpfad* (target path) genannt, angeben. Die folgenden Aktionen haben einen oder mehrere Zielpfade als Argumente:

- `loadMovie`
- `loadVariables`
- `unloadMovie`
- `setProperty`
- `startDrag`
- `duplicateMovieClip`
- `removeMovieClip`
- `print`
- `printAsBitmap`
- `tellTarget`

Die Aktion `loadMovie` erhält beispielsweise die Argumente *`URL`*, *`Location`* und *`Variables`*. *`URL`* ist die Web-Adresse des Filmes, den Sie laden möchten. *`Location`* ist der Zielpfad, in den der Film geladen werden soll.

```
loadMovie(URL, Location, Variables);
```

Anmerkung: Das Argument *`Variables`* wird in diesem Beispiel nicht benötigt.

Die folgende Anweisung lädt den URL http://www.mySite.com/myMovie.swf in die Instanz `bar` der Haupt-Zeitleiste. Der Zielpfad ist `_root; _root.bar`;

```
loadMovie("http://www.mySite.com/myMovie.swf", _root.bar);
```

In ActionScript wird eine Filmsequenz durch Ihren Instanznamen bezeichnet. In der folgenden Anweisung wird beispielsweise die Eigenschaft `_alpha` der Filmsequenz namens `star` auf 50% Sichtbarkeit gesetzt:

```
star._alpha = 50;
```

So geben Sie einer Filmsequenz einen Instanznamen:

1 Wählen Sie die Filmsequenz auf der Bühne aus.

2 Wählen Sie „Fenster“ > „Bedienfelder“ > „Instanz“.

3 Geben Sie im Feld „Name“ einen Instanznamen ein.

So identifizieren Sie einen geladenen Film:

Verwenden Sie `_levelX` wobei `X` die Nummer der Ebene ist, die in der Aktion `loadMovie`, die den Film geladen hat, angegeben wurde.

Ein Film, der beispielsweise in Ebene 5 geladen wurde, hat den Instanznamen `_level5`. Im folgenden Beispiel wird ein Film in Ebene 5 geladen und seine Sichtbarkeit auf „false“ gesetzt:

```
onClipEvent(load) {
    loadMovie("myMovie.swf", 5);
}
onClipEvent(enterFrame) {
   _level5._visible = false;
}
```

So geben Sie den Zielpfad eines Filmes ein:

Klicken Sie im Bedienfeld „Aktionen“ auf die Schaltfläche „Zielpfad“, und wählen Sie aus der angezeigten Liste eine Filmsequenz aus.

Weitere Informationen zum Schreiben von Zielpfaden finden Sie in Kapitel 4, „Arbeiten mit Filmsequenzen“.

So definieren Sie ein Ziel für einen Film mit Hilfe eines Ausdrucks:

Verwenden Sie die vordefinierte Funktion `targetPath` zum Auswerten des Ausdrucks, und verwenden Sie den zurückgegebenen Wert als Instanznamen.

So können Sie dynamisch Variablennamen erzeugen und Filmen dynamisch ein Ziel an einer beliebigen Stelle in der *Anzeigliste* (hierarchische Baumstuktur aller Sequenzen innerhalb eines Filmes) zuweisen. Es folgt ein Beispiel für die Verwendung von `targetPath`:

```
tellTarget(targetPath(Board.Block[index*2+1])) {
   play();
}
```

Falls Sie mit Flash 4 ActionScript vertraut sind: Die Verwendung von `targetPath` entspricht der Schrägstrich-Syntax:

```
tellTarget("Board/Block:" + (index*2+1)) {
   play();
}
```

Steuerung des Kontrollflusses in Skripts

In ActionScript werden die Aktionen `if`, `for`, `while`, `do...while` und `for...in` verwendet, wenn die Ausführung einer Aktion von einer Bedingung abhängt.

Verwenden von if-Anweisungen

Anweisungen, die überprüfen, ob eine Bedingung erfüllt ist (true) oder nicht (false), beginnen mit dem Schlüsselwort `if`. Wenn die Bedingung erfüllt ist, führt ActionScript die darauf folgende Anweisung aus. Wenn die Bedingung nicht erfüllt ist, springt ActionScript zur ersten Anweisung außerhalb des Codeblocks.

Überprüfen Sie zuerst die Bedingungen, die am häufigsten erfüllt sind. Dadurch wird die Ausführungsgeschwindigkeit des Codes optimiert.

Die folgenden Anweisungen überprüfen mehrere Bedingungen. Mit dem Schlüsselwort `else if` werden alternative Bedingungen angegeben, die überprüft werden, wenn alle vorherigen Bedingungen nicht erfüllt (false) sind.

```
if ((password == null) || (email == null)){
   gotoAndStop("reject");
}  else {
   gotoAndPlay("startMovie");
}
```

Wiederholen einer Aktion

In ActionScript kann eine Aktion wiederholt werden, wenn entweder eine Anzahl von Wiederholungen angegeben wurde oder für den Zeitraum, in dem eine festgelegte Bedingung erfüllt ist. Zum Erzeugen von Schleifen werden die Aktionen `while`, `do...while`, `for`, und `for...in` verwendet.

So wird eine Aktion wiederholt, während eine Bedingung erfüllt ist:

Verwenden Sie die `while`-Anweisung.

Eine `while`-Schleife wertet einen Ausdruck aus und führt den Code im Schleifenabschnitt so lange aus, wie der Ausdruck `true` (wahr) ist. Wenn alle Anweisungen im Schleifenabschnitt ausgeführt wurden, wird der Ausdruck erneut ausgewertet. Im folgenden Beispiel wird die Schleife viermal ausgeführt:

```
i = 4
while (i > 0) {
   myMC.duplicateMovieClip("newMC" +i, i );
   i --;
}
```

Mit einer `do...while`-Anweisung kann eine der `while`-Schleife ähnliche Schleife erzeugt werden. In einer `do...while`-Schleife wird der Ausdruck am Ende des Codeblocks ausgewertet, so dass die Schleife mindestens einmal ausgeführt wird. Ein Beispiel:

```
i = 4
do {
   myMC.duplicateMovieClip("newMC" +i, i );
   i --;
} while (i > 0);
```

So wiederholen Sie eine Aktion unter Verwendung eines eingebauten Zählers:

Verwenden Sie die `for`-Anweisung.

Die meisten Schleifen verwenden eine Art von Zähler, um zu steuern, wie oft die Schleife ausgeführt wird. Sie können eine Variable deklarieren und eine Anweisung schreiben, die den Wert bei jeder Ausführung der Schleife erhöht oder herabsetzt. In der Aktion `for` sind der Zähler und die Anweisung die den Zähler erhöht, Teil der Aktion, wie das folgende Beispiel zeigt:

```
for (i = 4; i > 0; i--){
   myMC.duplicateMovieClip("newMC" + i, i + 10);
}
```

So führen Sie ein Schleife über alle Unterelemente eines Filmes oder Objektes aus:

Verwenden Sie die `for..in`-Anweisung.

Unterelemente können andere Filmsequenzen, Funktionen, Objekte oder Variablen sein. Im folgenden Beispiel werden die Ergebnisse mit Hilfe von `trace` für den Druck im Ausgabefenster ausgegeben:

```
myObject = { Name:'Joe', Alter:25, Stadt:'San Francisco' };
for (propertyName in myObject) {
  trace("myObject hat die Eigenschaft: " + propertyName + ", mit
dem Wert: " + myObject[propertyName]);
}
```

Dieses Beispiel erzeugt im Ausgabefenster folgendes Ergebnis:

```
myObject hat die Eigenschaft: Name, mit dem Wert: Joe
myObject hat die Eigenschaft: Alter, mit dem Wert: 25
myObject hat die Eigenschaft: Stadt, mit dem Wert: San Francisco
```

Es ist auch möglich, das Skript nur über einen bestimmten Typ von Unterelement laufen zu lassen, beispielsweise nur über Unterelemente vom Typ Filmsequenz. Dies ist mit `for...in` in Verbindung mit dem Operator `typeof` möglich.

```
for (name in myMovieClip) {
  if (typeof (myMovieClip[name]) == "movieclip") {
    trace("Ich habe als Unterelement eine Filmsequenz namens " +
name);
  }
}
```

Anmerkung: Die `for..in`-Anweisung wiederholt ihre Aktion über alle Objekteigenschaften in der Prototyp-Kette des durchlaufenen Objektes. Wenn der Prototyp eines Unterobjektes `parent` ist, läuft `for..in` ebenfalls über die Eigenschaften von `parent`. Siehe „Erstellen von Vererbung“ auf Seite 87.

Weitere Informationen zu allen Aktionen finden Sie in den einzelnen Einträgen in Kapitel 7, „ActionScript-Verzeichnis“.

Verwenden von vordefinierten Funktionen

Eine Funktion ist ein Block von ActionScript-Code, der überall in einem Film wiederholt werden kann. Wenn einer Funktion bestimmte Werte, sogenannte Argumente, übergeben werden, arbeitet die Funktion mit diesen Werten. Eine Funktion kann auch Werte zurückgeben. Flash besitzt vordefinierte Funktionen, mit denen auf bestimmte Informationen zugegriffen und bestimmte Aufgaben durchgeführt werden können. So beispielsweise Kollisionserkennung (`hitTest`), der Wert der zuletzt gedrückten Taste (`keycode`) oder die Versionsnummer des Flash Players, auf dem der Film abgespielt wird (`getVersion`).

Aufrufen einer Funktion

Eine Funktion kann in einer beliebigen Zeitleiste aus einer beliebigen Zeitleiste, einschließlich eines geladenen Filmes, aufgerufen werden. Jede Funktion hat besondere Merkmale. Einigen Funktionen müssen bestimmte Werte übergeben werden. Wenn Sie einer Funktion mehr Argumente als erforderlich übergeben, werden die überflüssigen Werte ignoriert. Wenn Sie ein erforderliches Argument nicht übergeben, wird den leeren Argumenten der Datentyp `undefined` zugewiesen, was zu Fehlern beim Export des Skripts führen kann. Wenn eine Funktion aufgerufen werden soll, muss sie sich einem Bild befinden, das der Abspielkopf erreicht hat.

In der folgenden Tabelle sind die in Flash vordefinierten Funktionen aufgeführt:

Boolean	getTimer	isFinite	newline	scroll
escape	getVersion	isNaN	number	String
eval	globalToLocal	keycode	parseFloat	targetPath
false	hitTest	localToGlobal	parseInt	true
getProperty	int	maxscroll	random	unescape

Anmerkung: Von der Verwendung von Zeichenfolgen-Funktionen wird abgeraten. Sie sind in der obigen Liste nicht aufgeführt.

So rufen Sie eine Funktion im Expertenmodus auf:

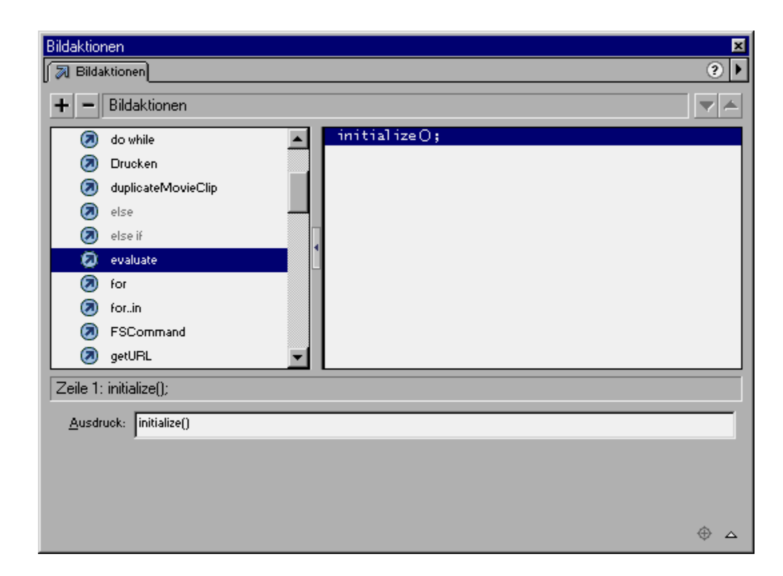
Verwenden Sie den Namen der Funktion. Übergeben Sie in Klammern alle erforderlichen Argumente.

Im folgenden Beispiel wird die Funktion `initialize` aufgerufen, die keine Argumente erfordert:

```
initialize();
```

So rufen Sie eine Funktion im Normalen Modus auf:

Verwenden Sie die Aktion `evaluate`. Geben Sie im Feld „Ausdruck“ den Namen der Funktion und alle erforderlichen Argumente ein.



Verwenden Sie zum Aufruf einer Funktion im Normalen Modus die Aktion „evaluate“.

Verwenden Sie zum Aufruf einer Funktion in einer anderen Zeitleiste einen Zielpfad. Zum Aufrufen der Funktion `calculateTax`, die in der Instanz `functionsMovieClip` definiert wurde, verwenden Sie beispielsweise folgenden Pfad:

```
_root.functionsMovieClip.calculateTax(total);
```

Anmerkung: Übergeben Sie alle Argumente innerhalb der Klammern.

Weitere Informationen zu allen Funktionen einschließlich der Zeichenfolgen-Funktionen, von deren Verwendung abgeraten wird, finden Sie in den einzelnen Einträgen in Kapitel 7, „ActionScript-Verzeichnis“.

Erstellen von benutzerdefinierten Funktionen

Sie können Funktionen definieren, um eine Reihe von Anweisungen mit übergebenen Werten ausführen zu lassen. Eine Funktion kann auch Werte zurückgeben. Nachdem eine Funktion definiert ist, kann sie aus jeder Zeitleiste heraus aufgerufen werden, einschließlich der Zeitleiste eines geladenen Filmes.

Eine Funktion kann als „Black Box“ betrachtet werden: Wenn eine Funktion aufgerufen wird, wird sie mit Eingaben (Argumenten) versorgt. Sie führt bestimmte Operationen durch und erzeugt eine Ausgabe (einen Rückgabewert). Eine korrekt geschriebene Funktion enthält umsichtig eingefügte Kommentare zu Eingabe, Ausgabe und Zweck. In diesem Falle muss der Anwender die Arbeitsweise der Funktion nicht in allen Einzelheiten verstehen.

Definieren einer Funktion

Funktionen sind, ebenso wie Variablen, an die Filmsequenz gebunden, in der sie definiert werden. Wenn eine Funktion erneut definiert wird, ersetzt die neue Definition die alte.

Sie definieren eine Funktion mit der Aktion `function`, gefolgt vom Namen der Funktion, den ihr zu übergebenden Argumenten und den ActionScript-Anweisungen, die den Inhalt der Funktion bilden.

Es folgt eine Funktion namens `Circle` mit dem Argument `radius`:

```
function Circle(radius) {
   this.radius = radius;
   this.area = Math.PI * radius * radius;
}
```

Anmerkung: Das Schlüsselwort `this`, das im Körper von Funktionen verwendet wird, ist eine Referenz auf die Filmsequenz, zu der die Funktion gehört.

Funktionen können auch durch Erzeugung eines *Funktions-Literals* (function literal) definiert werden. Ein Funktions-Literal ist eine unbenannte Funktion, die nicht in einer Anweisung, sondern in einem Ausdruck definiert wird. Das folgende Beispiel zeigt, wie Sie mit Hilfe eines Funktions-Literals in einem einzigen Ausdruck eine Funktion definieren, ihren Wert zurückgeben lassen und diesen einer Variablen zuweisen können:

```
area = (function () {return Math.PI * radius *radius;})(5);
```

Übergeben von Argumenten an eine Funktion

Argumente sind die Elemente, mit denen eine Funktion ihren Code ausführt. (In diesem Buch sind die Begriffe *Argument* und *Parameter* austauschbar.) Die Funktion im folgenden Beispiel übernimmt die Argumente `initials` und `finalScore`:

```
function fillOutScorecard(initials, finalScore) {
   scorecard.display = initials;
   scorecard.score = finalScore;
}
```

Beim Aufruf der Funktion müssen der Funktion die erforderlichen Argumente übergeben werden. Die Funktion ersetzt die Argumente in der Funktionsdefinition durch die übergebenen Werte. In diesem Beispiel ist `scorecard` der Instanzname einer Filmsequenz, `display` und `score` sind editierbare Textfelder innerhalb der Instanz. Der folgende Funktionsaufruf weist der Variablen `display` den Wert `"JEB"` und der Variablen `score` den Wert `45000` zu:

```
fillOutScorecard("JEB", 45000);
```

Das Argument `initials` der Funktion `fillOutScorecard` ähnelt einer lokalen Variablen. Es ist nur so lange vorhanden, wie die Funktion aufgerufen ist. Wenn Sie bei einem Funktionsaufruf Argumente weglassen, werden die weggelassenen Argumente als `undefined` übergeben. Wenn Sie bei einem Funktionsaufruf zusätzliche Argumente übergeben, die die Funktion laut Deklaration nicht benötigt, werden diese ignoriert.

Verwenden von lokalen Variablen in einer Funktion

Lokale Variablen sind wichtige Hilfsmittel, die Code strukturieren und leichter nachvollziehbar machen können. Wenn in einer Funktion lokale Variablen verwendet werden, sind diese Variablen vor allen anderen Skripts des Filmes verborgen. Der Gültigkeitsbereich von lokalen Variablen erstreckt sich auf den Körper der Funktion. Lokale Variablen werden beim Beenden der Funktion zerstört. Argumente, die der Funktion übergeben werden, werden ebenfalls als lokale Variablen behandelt.

Anmerkung: Wenn Sie in einer Funktion eine globale Variable ändern, sollten Sie Kommentare in das Skript einfügen, die diese Änderungen dokumentieren.

Zurückgeben von Werten aus einer Funktion

Funktionswerte werden mit der Aktion `return` zurückgegeben. Die Aktion `return` beendet die Funktion und ersetzt sie durch den Werte der Aktion `return`. Wenn Flash vor dem Ende einer Funktion keine Aktion `return` vorfindet, wird eine leere Zeichenfolge zurückgegeben. Die Funktion im folgenden Beispiel gibt das Quadrat des Arguments `x` zurück:

```
function sqr(x) {
   return x * x;
}
```

Manche Funktionen führen eine Reihe von Aufgaben durch, ohne einen Wert zurückzugeben. Die Funktion im folgendem Beispiel initialisiert eine Reihe von globalen Variablen:

```
function initialize() {
   boat_x = _root.boat._x;
   boat_y = _root.boat._y;
   car_x = _root.car._x;
   car_y = _root.car._y;
}
```

Aufrufen einer Funktion

Aus dem Bedienfeld „Aktionen“ im Normalen Modus werden Funktion mit der Aktion `evaluate` gestartet. Übergeben Sie in Klammern alle erforderlichen Argumente. Eine Funktion in einer beliebigen Zeitleiste kann aus einer beliebigen Zeitleiste, einschließlich eines geladenen Filmes, aufgerufen werden. Die nachfolgende Anweisung ruft beispielsweise die Funktion `sqr` in der Filmsequenz `MathLib` in der Hauptzeitleiste auf, übergibt ihr das Argument 3 und speichert das Resultat in der Variablen `temp`:

```
var temp = _root.MathLib.sqr(3);
```

Um in Flash 4 den Aufruf einer Funktion zu simulieren, könnten Sie ein Skript für ein Bild nach Ende des Filmes schreiben und es starten, indem Sie der Aktion `call` den Namen der Bildbezeichnung übergeben. Wenn ein Skript, das Variablen initialisiert hat, sich beispielsweise in einem Bild mit der Bezeichnung `initialize` befindet, müssen Sie es wie folgt aufrufen:

```
call("initialize");
```

Diese Art von Skript war keine echte Funktion, da keine Argumente übergeben werden konnten und das Skript keine Werte zurückgeben konnte. Obwohl die Aktion `call` in Flash 5 noch verfügbar ist, wird von ihrer Verwendung abgeraten.

Verwenden von vordefinierten Objekten

Die in Flash vordefinierten Objekte ermöglichen den Zugriff auf bestimmte Arten von Informationen. Die meisten vordefinierten Objekte besitzen *Methoden* (Funktionen, die zu einem Objekt gehören), die beim Aufruf einen Wert zurückgeben oder eine Aktion durchführen. Das Date-Objekt gibt beispielsweise Informationen der Systemuhr zurück, und über das Sound-Objekt können Soundelemente ihres Filmes gesteuert werden.

Einige vordefinierte Objekte besitzen Eigenschaften, deren Werte gelesen werden können. Das Key-Objekt besteht beispielsweise aus konstanten Werten, die die Tasten der Tastatur darstellen. Jedes Objekt besitzt besondere Merkmale und Fähigkeiten, die Sie in dem Film einsetzen können.

Es folgt eine Aufzählung der in Flash vordefinierten Objekte:

- Array
- Boolean
- Color
- Date
- Key
- Math
- MovieClip
- Number
- Object
- Selection
- Sound
- String
- XML
- XMLSocket

Instanzen von Filmsequenzen werden in ActionScript als Objekte dargestellt. Vordefinierte Methoden von Filmsequenzen werden wie Methoden eines beliebigen anderen ActionScript-Objektes aufgerufen.

Ausführliche Informationen zu allen Objekten finden Sie im entsprechenden Eintrag in Kapitel 7, „ActionScript-Verzeichnis“.

Erstellen eines Objekts

Ein Objekt kann auf zwei Arten erstellt werden: Mit dem Operator `new` oder mit dem Objektinitialisierungsoperator (`{}`). Mit dem Operator `new` können sowohl Objekte einer vordefinierten Objektklasse als auch Objekte einer benutzerdefinierten Objektklasse erstellt werden. Mit dem Objektinitialisierungsoperator (`{}`) können Objekte des generischen Typs Object erstellt werden.

Wenn der Operator `new` zum Erzeugen eines Objektes verwendet wird, muss er zusammen mit einer Konstruktor-Funktion verwendet werden. (Eine Konstruktor-Funktion ist einfach eine Funktion, deren einzige Aufgabe darin besteht, einen bestimmten Objekttyp zu erzeugen.) Die in ActionScript vordefinierten Objekte sind im Grunde vordefinierte Konstruktor-Funktionen. Das neue Objekt *instantiiert* (erstellt) eine Kopie des Objektes und weist der Kopie alle Eigenschaften und Methoden des Objektes zu. Dies ähnelt dem Ziehen einer Filmsequenz aus der Bibliothek auf die Bühne. Die Anweisungen im folgenden Beispiel instantiieren ein Objekt vom Typ Date:

```
currentDate = new Date();
```

Bei einigen vordefinierten Objekten sind die Methoden ohne Instantiierung zugänglich. In der folgenden Anweisung wird beispielsweise die Methode `random` des Objektes Math aufgerufen:

```
Math.random();
```

Jedes Objekt, das eine Konstruktor-Funktion benötigt, verfügt über ein entsprechendes Element in der Werkzeugleiste des Bedienfeldes „Aktionen“, beispielsweise `new Color`, `new Date`, `new String` usw.

So erstellen Sie im Normalen Modus ein Objekt mit dem Operator `new`**:**

1. Wählen Sie `setVariable`
2. Geben Sie im Feld „Name“ einen Variablennamen ein.
3. Geben Sie im Feld „Wert“ `new Object`, `new Color` usw. ein. Geben Sie in Klammern die von der Konstruktor-Funktion benötigten Argumente ein.
4. Aktivieren Sie im Feld „Wert“ das Kontrollkästchen „Ausdruck“.

   Wenn das Kontrollkästchen „Ausdruck“ nicht aktiviert ist, wird der gesamte Wert als Zeichenfolgenliteral interpretiert.

Im nachfolgenden Code wird das Objekt `c` mit dem Konstruktor Color erstellt:

```
c = new Color(this);
```

Anmerkung: Ein Objektname ist eine Variable, der der Datentyp Object zugewiesen ist.

So greifen Sie im Normalen Modus auf eine Methode zu:

1 Wählen Sie die Aktion `evaluate` aus.

2 Geben Sie im Feld „Ausdruck“ den Namen des Objektes ein.

3 Geben Sie im Feld „Ausdruck“ eine Eigenschaft des Objektes ein.

So verwenden Sie den Objektinitialisierungsoperator ({}) im Normalen Modus:

1 Wählen Sie die Aktion `setVariable` aus.

2 Geben Sie den Namen des neuen Objektes im Feld „Variable“ ein.

3 Geben Sie Eigenschaftenname und -wert paarweise durch einen Doppelpunkt getrennt innerhalb des Objektinitialisierungsoperators ({}) ein.

In dieser Anweisung sind beispielsweise `radius` und `area` die Namen der Eigenschaften und `5` und der Wert eines Ausdrucks sind deren Werte:

```
myCircle = {radius: 5, area:(pi * radius * radius)};
```

Die Klammern bewirken die Auswertung des Ausdrucks. Der zurückgegebene Wert ist der Wert der Variablen `area`.

Initialisierungen von Arrays und Objekten können auch verschachtelt werden:

```
newObject = {name: "John Smith", projects: ["Flash", "Dream-
weaver"]};
```

Ausführliche Informationen zu allen Objekten finden Sie im entsprechenden Eintrag in Kapitel 7, „ActionScript-Verzeichnis“.

Zugreifen auf Objekteigenschaften

Der Zugriff auf Werte von Eigenschaften eines Objektes erfolgt durch den Punkt-Operator (.). Auf der linken Seite steht der Name des Objektes und auf der rechten Seite der Name der Eigenschaft. Im nachfolgenden Beispiel ist `myObject` das Objekt und `name` die Eigenschaft:

```
myObject.name
```

Im Normalen Modus werden Eigenschaften Werte mit der Aktion `setVariable` zugewiesen:

myObject.name = „Allen“

Der Wert einer Eigenschaft wird geändert, indem ihr ein neuer Wert zugewiesen wird. Ein Beispiel:

```
myObject.name = "Homer";
```

Sie können auf Eigenschaften eines Objektes auch mit dem Arrayzugriffsoperator ([]) zugreifen. Siehe „Punkt-Operator und Array-Zugriffsoperator“ auf Seite 70.

Aufrufen von Objekt-Methoden

Eine Methode wird aufgerufen, indem die Methode dem Punkt-Operator nachgestellt wird. Im folgenden Beispiel wird die Methode `setVolume` des Sound-Objektes aufgerufen:

```
s = new Sound(this);
s.setVolume(50);
```

Im Normalen Modus wird eine Methode eines vordefinierten Objektes mit der Aktion `evaluate` aufgerufen.

Verwenden des MovieClip-Objekts

Die Methoden des vordefinierten MovieClip-Objektes dienen zur Steuerung der Instanzen von Filmsequenzsymbolen auf der Bühne. Im folgenden Beispiel wird das Abspielen der Instanz `dateCounter` veranlasst:

```
dateCounter.play();
```

Ausführliche Informationen zum MovieClip-Objekt finden Sie im entsprechenden Eintrag in Kapitel 7, „ActionScript-Verzeichnis“.

Verwenden des Array-Objekts

Das Array-Objekt ist ein häufig verwendetes vordefiniertes ActionScript-Objekt, das seine Daten in durchnummerierten Eigenschaften anstelle von mit Namen bezeichneten Eigenschaften speichert. Der Name eines Array-Elements wird als *Index* bezeichnet. Dies erleichtert das Speichern und Abrufen bestimmter Informationen, beispielsweise Listen von Personen oder die Abfolge von Zügen in einem Spiel.

Zuweisungen an Elemente des Array-Objektes erfolgen wie Zuweisungen an eine Eigenschaft eines beliebigen Objekts:

```
move[1] = "a2a4";
move[2] = "h7h5";
move[3] = "b1c3";
...
move[100] = "e3e4";
```

Mit dem Ausdruck `move[2]` wird auf das zweite Element des Arrays zugegriffen.

Das Array-Objekt besitzt eine vordefinierte Eigenschaft `length` (Länge), die die Anzahl der Elemente des Arrays als Wert hat. Wenn bei einer Zuweisung an ein Element des Array-Objektes der Index des Elements positiv ist und `Index >= length` gilt, wird `length` automatisch auf `Index + 1` aktualisiert.

Verwenden von benutzerdefinierten Objekten

Sie können benutzerdefinierte Objekte erzeugen, um die Informationen in Ihren Skripts im Hinblick auf Speicherung und Zugriff durch die Definition von Eigenschaften und Methoden besser zu strukturieren. Nach der Erzeugung eines Master-Objektes ("Klasse") können sie Kopien (d. h. Instanzen) dieses Objektes in einem Film verwenden ("instantiieren"). Dadurch können Sie Code wiederverwenden und Speicherplatz in Dateien einsparen.

Ein Objekt ist ein komplexer Datentyp, der 0 oder mehr Eigenschaften besitzt. Jede Eigenschaft hat, ebenso wie eine Variable, einen Namen und einen Wert. Die Eigenschaften sind an das Objekt gebunden und enthalten Werte, die geändert und abgerufen werden können. Diese Werte können einen beliebigen Datentyp haben: Zeichenfolge, Zahl, Boolean, Objekt, Filmsequenz oder `undefined`. Die folgenden Eigenschaften haben die verschiedene Datentypen:

```
customer.name = "Jane Doe"
customer.age = 30
customer.member = true
customer.account.currentRecord = 000609
customer.mcInstanceName._visible = true
```

Eine Eigenschaft eines Objektes kann wieder ein Objekt sein. In Zeile 4 des vorigen Beispiels ist `account` eine Eigenschaft des Objektes `customer` und `currentRecord` eine Eigenschaft des Objektes `account`. Der Datentyp der Eigenschaft `currentRecord` ist Zahl.

Erstellen eines Objekts

Zum Erstellen eines Objektes durch eine Konstruktor-Funktion wird der Operator `new` verwendet. Eine Konstruktor-Funktion trägt immer den Namen des Objektes, das sie erzeugt. Ein Konstruktor, der beispielsweise ein Objekt vom Typ Account erzeugt, hieße `Account`. In der folgenden Anweisung wird ein neues Objekt durch die Funktion namens `MyConstructorFunction` erzeugt:

```
new MyConstructorFunction (argument1, argument2, ... argumentN);
```

Beim Aufrufen von `MyConstructorFunction` übergibt Flash das versteckte Argument `this`, das eine Referenz auf das von `MyConstructorFunction` zu erzeugende Objekt darstellt. Innerhalb eines Konstruktors ermöglicht `this` Bezüge auf das Objekt, das vom Konstruktor erzeugt wird. Das folgende Beispiel zeigt eine Konstruktor-Funktion, die einen Kreis erzeugt:

```
function Circle(radius) {
  this.radius = radius;
  this.area = Math.PI * radius * radius;
}
```

Konstruktor-Funktionen werden normalerweise zum Einsetzen von Methoden des Objektes verwendet.

```
function Area() {
   this.circleArea = MAth.PI * radius * radius;
}
```

Um ein Objekt in einem Skript verwenden zu können, müssen Sie ihm einen Namen zuweisen. Sie erzeugen ein neues Objekt vom Typ Kreis mit Radius 5, in dem Sie das Objekt mit dem Operator `new` erzeugen und es der lokalen Variablen `myCircle` zuweisen:

```
var myCircle = new Circle(5);
```

Anmerkung: Objekte haben denselben Gültigkeitsbereich wie die Variable, der sie zugewiesen werden. Siehe „Festlegen des Gültigkeitsbereichs einer Variablen“ auf Seite 61.

Erstellen von Vererbung

Jede Funktion besitzt eine Eigenschaft namens `prototype`, die automatisch bei der Definition der Funktion erzeugt wird. Beim Erzeugen eines neuen Objektes durch eine Konstruktor-Funktion werden sämtliche Eigenschaften und Methoden der Eigenschaft `prototype` des Konstruktors zu Eigenschaften und Methoden der Eigenschaft `__proto__` des neuen Objektes. Die Eigenschaft `prototype` beinhaltet die Standard-Eigenschaftenwerte für Objekte, die mit dieser Funktion erzeugt werden. Die Übergabe von Werten zwischen den Eigenschaften `__proto__` und `prototype` wird als Vererbung bezeichnet.

Die Vererbung erfolgt nach einer festgelegten Hierarchie. Wenn Sie eine Eigenschaft oder eine Methode eines Objektes aufrufen, prüft ActionScript, ob ein derartiges Element vorhanden ist. Wenn es nicht vorhanden ist, sucht ActionScript in der Eigenschaft `__proto__` des Objektes nach dieser Information (`object.__proto__`). Wenn die Eigenschaft, auf die zugegriffen werden soll, keine Eigenschaft des `__proto__` -Objektes des Objektes ist, sucht ActionScript in `object.__proto__.__proto__`.

Es ist üblich, Methoden an ein Objekt durch Zuweisung der Methoden an die Eigenschaft `prototype` des Objektes zu binden. Die folgenden Schritte beschreiben die Definition einer Beispielmethode:

1 Die Konstruktur-Funktion `Circle` wird wie folgt definiert:

```
function Circle(radius) {
  this.radius = radius
}
```

2 Dann wird die Methode `area` des Objektes `Circle` definiert. Die Methode `area` berechnet den Flächeninhalt des Kreises. Im folgenden Beispiel wird die Methode `area` durch ein Funktionsliteral definiert und die Eigenschaft area des prototype-Objektes von Circle gesetzt:

```
Circle.prototype.area = function () {
     return Math.PI * this.radius * this.radius
   }
```

3 Dann wird eine Instanz des Objektes `Circle` erzeugt:

```
var myCircle = new Circle(4);
```

4 Dann wird die Methode `area` des neuen Objektes `myCircle` aufgerufen:

```
var myCircleArea = myCircle.area()
```

ActionScript sucht im Objekt `myCircle` nach der Methode `area`. Da das Objekt keine Methode namens `area` besitzt, wird sein prototype-Objekt `Circle.prototype` nach der Methode `area` durchsucht. ActionScript findet sie und ruft sie auf.

Eine Methode kann auch an ein Objekt gebunden werden, indem die Methode an jede einzelne Instanz des Objektes gebunden wird. Ein Beispiel:

```
function Circle(radius) {
  this.radius = radius
  this.area = function() {
    return Math.PI * this.radius * this.radius
  }
}
```

Von dieser Technik ist abzuraten. Die Verwendung des `prototype`-Objektes ist effizienter, da nur eine Definition von `area` erforderlich ist. Diese Definition wird automatisch in alle von der Funktion `Circle` erzeugte Objekte kopiert.

Die Eigenschaft `prototype` wird von Flash Player ab Version 5 unterstützt. Weitere Informationen finden Sie in Kapitel 7, „ActionScript-Verzeichnis“.

Öffnen von Flash 4 Dateien

ActionScript hat sich mit Erscheinen von Flash 5 erheblich verändert. Es ist nun eine objektorientierte Sprache mit mehreren Datentypen und Punkt-Sytax. In Flash 4 ActionScript stand nur ein eigentlicher Datentyp zur Verfügung: Zeichenfolge. In Ausdrücken wurden unterschiedliche Typen von Operatoren verwendet, um festzulegen, ob der Wert als Zeichenfolge oder Zahl behandelt werden sollte. In Flash 5 können Sie denselben Satz von Operatoren auf alle Datentypen anwenden.

Wenn Sie eine mit Flash 4 erzeugte Datei mit Flash 5 öffnen, konvertiert Flash automatisch ActionScript-Ausdrücke, um sie mit der Flash 5-Syntax kompatibel zu machen. Folgende Datentyp- und Operator-Konversionen werden Ihnen im ActionScript-Code begegnen:

- Der Operator = wurde in Flash 4 für numerische Gleichheit verwendet. In Flash 5 ist == der Gleichheitsoperator und = der Zuweisungsoperator. Jedes Vorkommen des Operators = in Flash 4-Dateien wird automatisch in == konvertiert.
- Flash führt automatisch Typkonvertierungen durch, um zu gewährleisten, dass sich die Operatoren wie erwartet verhalten. Seit der Einführung von unterschiedlichen Datentypen haben die folgenden Operatoren neue Bedeutungen:

  ```
  +, ==, !=, <>, <, >, >=, <=
  ```

- Diese Operatoren waren in Flash 4 ActionScript immer numerische Operatoren. In Flash 5 verhalten sie sich in Abhängigkeit von den Datentypen der Operanden unterschiedlich. Um bei importierten Dateien semantischen Unterschieden vorzubeugen, wird für alle Operanden dieser Operatoren ein Aufruf der Funktion `Number` eingefügt. (Zahlenkonstanten sind selbstverständlich bereits Zahlen und werden daher nicht in einen Aufruf der Funktion `Number` eingeschlossen.)
- In Flash 4 erzeugte die Escape-Sequenz `\n` ein Wagenrücklauf-Zeichen (Carriage Return, ASCII 13). In Flash 5 erzeugt `\n` entsprechend dem ECMA-262-Standard ein Zeilenvorschub-Zeichen (Linefeed, ASCII 10). Die Sequenz `\n` in Flash 4-FLA-Dateien wird automatisch in `\r` konvertiert.
- Der Operator `&` wurde in Flash 4 zur Addition von Zeichenfolgen verwendet. In Flash 5 ist `&` der bitweise UND-Operator. Der Operator für die Zeichenfolgen-Addition heißt nun `add`. Jedes Vorkommen des Operators `&` in Flash 4-Dateien wird automatisch in den Operator `add` konvertiert.
- Viele Funktionen benötigten in Flash 4 keine schließenden Klammern, beispielsweise `Get Timer`, `Set Variable`, `Stop` und `Play`. Im Sinne einer konsistenten Syntax benötigen die Funktion `getTimer` und alle Aktionen in Flash 5 schließende Klammern. Diese Klammern werden bei der Konvertierung automatisch hinzugefügt.
- Wenn die Funktion `getProperty` für eine nicht vorhandene Filmsequenz ausgeführt wird, gibt sie in Flash 5 nicht 0, sondern den Wert `undefined` zurück. `undefined == 0` ist in Flash 5 ActionScript `false` (falsch). Flash löst bei der Konvertierung von Flash 4-Dateien dieses Problem durch Aufruf der Funktion `Number` bei Vergleichen auf Gleichheit. Im folgenden Beispiel bewirkt `Number`, dass `undefined` in 0 konvertiert wird, so dass der Vergleich erfolgreich ist:

  ```
  getProperty("clip", _width) == 0
  Number(getProperty("clip", _width)) == Number(0)
  ```

Anmerkung: Wenn Sie in Flash 4 ActionScript ein Flash 5-Schlüsselwort als Variablennamen verwendet haben, führt dies zu einem Syntaxfehler. Zur Behebung dieses Problems müssen Sie die Variablen an allen Stellen umbenennen. Siehe „Schlüsselwörter“ auf Seite 55.

Erzeugen von Flash 4-Inhalten mit Flash 5

Wenn Sie mit Flash 5 Inhalte für Flash 4 Player (durch Exportieren als Flash 4) erstellen, können Sie nicht alle neuen Funktionen von Flash 5 ActionScript nutzen. Viele der neuen Funktionen von ActionScript sind jedoch weiterhin verfügbar. Flash 4 ActionScript besitzt nur einen Grunddatentyp, der für die Verarbeitung von Zahlen und Zeichenfolgen eingesetzt wird. Wenn Sie einen Film für Flash 4 Player erstellen, müssen Sie die veralteten Zeichenfolgenoperatoren aus der Kategorie „Zeichenfolgenoperatoren" der Werkzeugleiste verwenden.

Sie können folgende Flash 5-Funktionen verwenden, wenn Sie in das Dateiformat Flash 4 SWF exportieren:

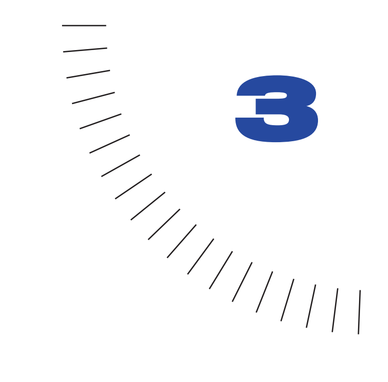
- Zugriffsoperator für Arrays und Objekte (`[]`).
- Punkt-Operator (`.`).
- Logische Operatoren, Zuweisungsoperatoren, Prä-Inkrement/Dekrement und Post-Inkrement/Dekrement-Operatoren.
- Modulo-Operator (`%`), alle Methoden und Eigenschaften des Math-Objektes.

  Diese Operatoren und Funktionen werden von Flash 4 Player nicht unterstützt. Flash 5 muss sie als Approximation durch Reihenentwicklung exportieren. Dies bedeutet, dass die Ergebnisse nur Annäherungen sind. Darüber hinaus beanspruchen diese Funktionen in Flash 4 SWF-Dateien mehr Speicherplatz als in Flash 5 SWF-Dateien, da die Reihenentwicklungen in die SWF-Datei aufgenommen werden müssen.
- Aktionen `for`, `while`, `do while`, `break` und `continue`.
- Aktionen `print` und `printAsBitmap` .

Die folgenden Flash 5-Funktionen können nicht in Filmen verwendet werden, die im Flash 4 SWF-Dateiformat exportiert werden:

- Benutzerdefinierte Funktionen
- XML-Unterstützung
- Lokale Variablen
- Vordefinierte Objekte (außer Math)
- Filmsequenzaktionen
- Mehrere Datentypen
- `eval` mit Punkt-Syntax (beispielsweise `eval ("_root.movieclip.variable")`)
- `return`
- `new`
- `delete`

- typeof
- for..in
- keycode
- targetPath
- escape
- globalToLocal und localToGlobal
- hitTest
- isFinite und inNaN
- parseFloat und parseInt
- tunescape
- _xmouse und _ymouse
- _quality



KAPITEL 3
Erzeugen von Interaktion mit ActionScript

Bei einem interaktiven Film werden die Zuschauer einbezogen. Mit Hilfe der Tastatur und/oder der Maus kann der Zuschauer zu bestimmten Teilen des Filmes springen, Objekte verschieben, auf Schaltflächen klicken und andere interaktive Operationen durchführen.

Interaktive Filme werden mit Hilfe von Skripts erstellt, die ausgeführt werden, sobald ein bestimmtes Ereignis eintritt. Ereignisse, die ein Skript auslösen können, liegen z. B. dann vor, wenn ein bestimmtes Bild abgespielt wird, eine Filmsequenz geladen oder entladen wird oder der Benutzer auf eine Schaltfläche klickt oder eine Taste auf der Tastatur drückt. Mit ActionScript erstellen Sie Skripts, die Flash mitteilen, welche Aktion beim Eintreten eines Ereignisses durchgeführt werden soll.

Die folgenden grundlegenden Aktionen stellen übliche Verfahren dar, um Bewegung und Benutzerinteraktion in einem Film zu steuern:

- Abspielen und Anhalten von Filmen
- Einstellen der Filmqualität
- Deaktivieren aller Sounds
- Springen zu einem Bild bzw. einer Szene
- Springen zu einem anderen URL
- Überprüfen, ob ein Bild geladen wurde
- Laden und Entladen zusätzlicher Filme

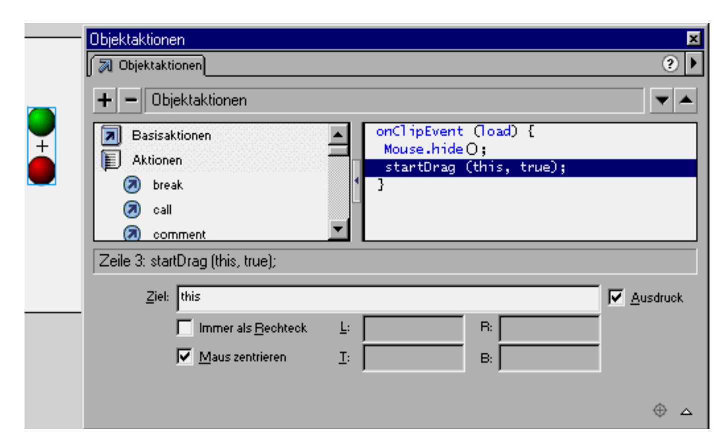
Ausführliche Informationen zu diesen Aktionen finden Sie im *Flash Benutzerhandbuch*.

Wenn Sie weitergehende Interaktivität realisieren möchten, müssen Sie in der Lage sein, die folgenden Techniken einzusetzen:

- Erstellen eines benutzerdefinierten Mauszeigers
- Abfragen der Mausposition
- Abfangen von Tastendrücken
- Erstellen eines Textfeldes mit Bildlauf
- Setzen von Farbwerten
- Erstellen von Sound-Steuerelementen
- Erkennen von Kollisionen

Erstellen eines benutzerdefinierten Mauszeigers

Der Standard-Mauszeiger (d. h. die Bildschirmdarstellung des Mauszeigers) wird mit Hilfe der Methode `hide` des vordefinierten Mouse-Objektes ausgeblendet. Mit der Aktion `startDrag` können Sie eine Filmsequenz als benutzerdefinierten Mauszeiger verwenden.



Mit einer Filmsequenz verbundene Aktionen zur Erstellung eines benutzerdefinierten Mauszeigers

So erstellen Sie einen benutzerdefinierten Mauszeiger:

1. Erstellen Sie eine Filmsequenz, die Sie als benutzerdefinierten Mauszeiger verwenden möchten.
2. Wählen Sie die Instanz der Filmsequenz auf der Bühne aus.

3 Wählen Sie „Fenster“ > „Aktionen“, und öffnen Sie das Bedienfeld „Objektaktionen“.

4 Wählen Sie in der Werkzeugliste zuerst „Objekte“ und dann „Maus“ aus, und ziehen Sie `Ausblenden` in das Skriptfenster.

   Dabei entsteht folgender Code:

```
onClipEvent(load){
    Mouse.hide();
}
```

5 Wählen Sie in der Werkzeugliste „Aktionen“ aus, und ziehen Sie `startDrag` in das Skriptfenster.

6 Aktivieren Sie das Kontrollkästchen „Maus zentrieren“.

   Dabei entsteht folgender Code:

```
onClipEvent(load){
    Mouse.hide()
    startDrag(this, true);
}
```

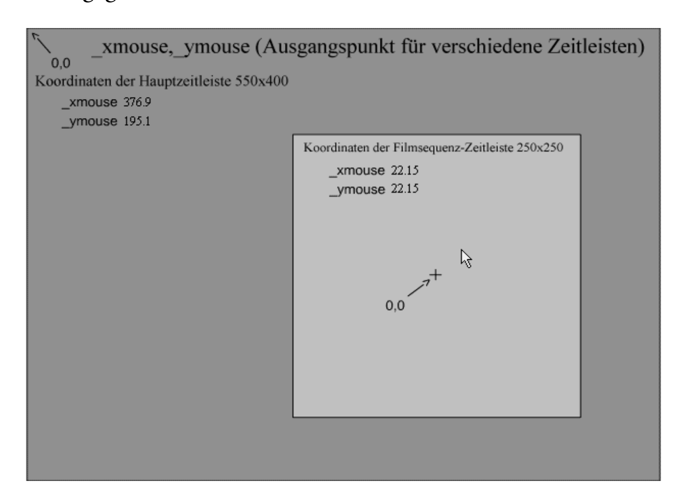
7 Wählen Sie „Steuerung“ > „Film testen“, um den benutzerdefinierten Mauszeiger zu verwenden.

Wenn Sie einen benutzerdefinierten Mauszeiger einsetzen, funktionieren Schaltflächen wie bisher. Es ist empfehlenswert, den benutzerdefinierten Mauszeiger auf die oberste Ebene der Zeitleiste zu setzen, damit er sich vor Schaltflächen und anderen Objekten bewegt, wenn Sie die Maus innerhalb des Filmes bewegen.

Weitere Informationen zu den Methoden des Mouse-Objektes finden Sie in Kapitel 7, „Referenz zu ActionScript“.

Abfragen der Mausposition

Die _xmouse- und _ymouse-Eigenschaften liefern die Position des Mauszeigers innerhalb eines Filmes. Jede Zeitleiste besitzt die _xmouse- und _ymouse-Eigenschaften, mit denen die Position der Maus innerhalb ihres Koordinatensystems zurückgegeben wird.



Die _xmouse- und _ymouse-Eigenschaften innerhalb der Hauptzeitleiste und der Zeitleiste einer Filmsequenz

Die folgende Anweisung, die die _xmouse-Position der Maus innerhalb der Hauptzeitleiste ermittelt, könnte in jeder Zeitleiste des Filmes _level0 stehen:

```
x_pos = _root._xmouse;
```

Verwenden Sie den Instanznamen der Filmsequenz, wenn Sie die Mausposition innerhalb der Filmsequenz ermitteln möchten. Die folgende Anweisung, die die _ymouse-Position in der Instanz myMovieClip ermittelt, könnte beispielsweise in jeder Zeitleiste des Filmes _level0 stehen:

```
y_pos = _root.myMovieClip._ymouse
```

Sie können die Mausposition innerhalb einer Filmsequenz auch ermitteln, indem Sie die „_xmouse“- und „_ymouse“-Eigenschaften in einer Filmsequenzaktion einsetzen, wie im folgenden Beispiel:

```
onClipEvent(enterFrame){
   xmousePosition = _xmouse;
   ymousePosition = _ymouse;
}
```

Die Variablen `x_pos` und `y_pos` dienen zum Speichern der Mauspositionen. Sie können diese Variablen in einem beliebigen Skript Ihres Filmes verwenden. Im folgenden Beispiel werden die Werte von `x_pos` und `y_pos` immer dann aktualisiert, wenn der Benutzer die Maus bewegt.

```
onClipEvent(mouseMove){
   x_pos = _root._xmouse;
   y_pos = _root._ymouse;
}
```

Weitere Informationen zu den `_xmouse`- und `_ymouse`-Eigenschaften finden Sie in Kapitel 7, „Referenz zu ActionScript“.

Abfangen von Tastendrücken

Mit den Methoden des vordefinierten Key-Objektes kann die vom Benutzer zuletzt gedrückte Taste ermittelt werden. Für das Key-Objekt wird keine Konstruktorfunktion benötigt. Wie das folgende Beispiel zeigt, rufen Sie einfach das Objekt selbst auf, wenn Sie seine Methoden verwenden möchten:

```
Key.getCode();
```

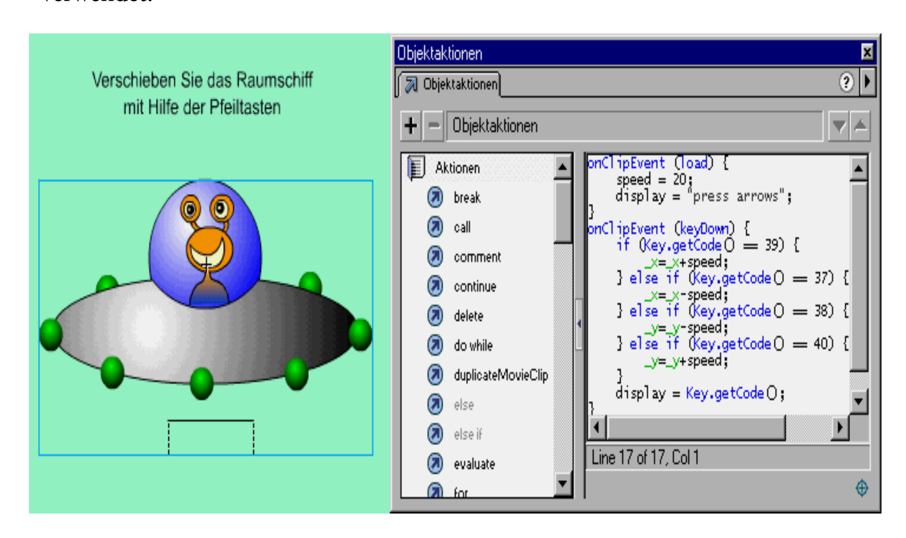
Sie können virtuelle Tastencodes oder die ASCII-Werte der gedrückten Tasten ermitteln:

- Den virtuellen Tastencode der zuletzt gedrückten Taste erhalten Sie mit der Methode `getCode`.
- Den ASCII-Wert der zuletzt gedrückten Taste erhalten Sie mit der Methode `getAscii`.

Jeder physischen Taste der Tastatur ist ein virtueller Tastencode zugeordnet. Die NACH-LINKS-TASTE weist z. B den virtuellen Tastencode 37 auf. Wenn Sie virtuelle Tastencodes verwenden, können Sie sicher sein, dass die Steuerelemente Ihres Filmes unabhängig von Sprache und Plattform arbeiten.

Den ersten 127 Zeichen eines jeden Zeichensatzes sind ASCII-Werte (American Standard Code for Information Interchange) zugeordnet. ASCII-Werte liefen Informationen über Zeichen auf dem Bildschirm. Die Buchstaben „A“ und „a“ weisen z. B verschiedene ASCII-Werte auf.

`Key.getCode` wird normalerweise in einem `onClipEvent`-Handler eingesetzt. Durch die Übergabe von `keyDown` als Parameter weist der Handler ActionScript an, den Wert der zuletzt gedrückten Taste nur dann zu ermitteln, wenn gerade eine Taste gedrückt wurde. In diesem Beispiel wird `Key.getCode` in einer `if`-Anweisung zum Erzeugen von Steuerelementen zur Navigation des Raumschiffs verwendet.



So erzeugen Sie Tastatur-Steuerelemente für einen Film:

1 Legen Sie fest, welche Tasten verwendet werden sollen, und bestimmen Sie mit einer der folgenden Methoden deren virtuellen Tastencode:

- Eine Liste der Tastencodes finden Sie in Anhang B, „Tastatureingaben und Tastencodewerte".
- Verwenden Sie eine Konstante des Key-Objektes. (Wählen Sie in der Werkzeugliste erst „Objekte" und dann „Taste" aus. Konstanten werden in Großbuchstaben angezeigt.)
- Weisen Sie die folgende Filmsequenz-Aktion zu, wählen Sie dann „Steuerung" > „Film testen", und drücken Sie die gewünschte Taste:

```
onClipEvent(keyDown) {
    trace(Key.getCode());
}
```

2 Wählen Sie eine Filmsequenz auf der Bühne aus.

3 Wählen Sie „Fenster" > „Aktionen".

4 Doppelklicken Sie in der Kategorie „Aktionen" der Werkzeugleiste auf die Aktion `onClipEvent` .

5 Wählen Sie im Bedienfeld „Parameter" das Ereignis `Key down`.

6 Doppelklicken Sie in der Kategorie „Aktionen“ der Werkzeugleiste auf die Aktion `if`.

7 Klicken Sie auf den Parameter „Bedingung“, wählen Sie erst „Objekte“, anschließend „Taste“ und dann `getCode` aus.

8 Doppelklicken Sie in der Kategorie „Operatoren“ der Werkzeugleiste auf den Gleichheitsoperator (==).

9 Geben Sie rechts vom Gleichheitsoperator den virtuellen Tastencode ein.

   Dabei entsteht folgender Code:

```
onClipEvent(keyDown) {
    if (Key.getCode() == 32) {
    }
}
```

10 Wählen Sie die Aktion aus, die beim Drücken der festgelegten Taste ausgeführt werden soll.

   Die folgende Aktion bewirkt z. B, dass die Hauptzeitleiste zum nächsten Bild übergeht, wenn die LEERTASTE (32) gedrückt wird:

```
onClipEvent(keyDown) {
    if (Key.getCode() == 32) {
       nextFrame();
    }
}
```

Weitere Informationen zu den Methoden des Key-Objektes finden Sie in den entsprechenden Einträgen in Kapitel 7, „Referenz zu ActionScript“.

Erstellen eines Textfeldes mit Bildlauf

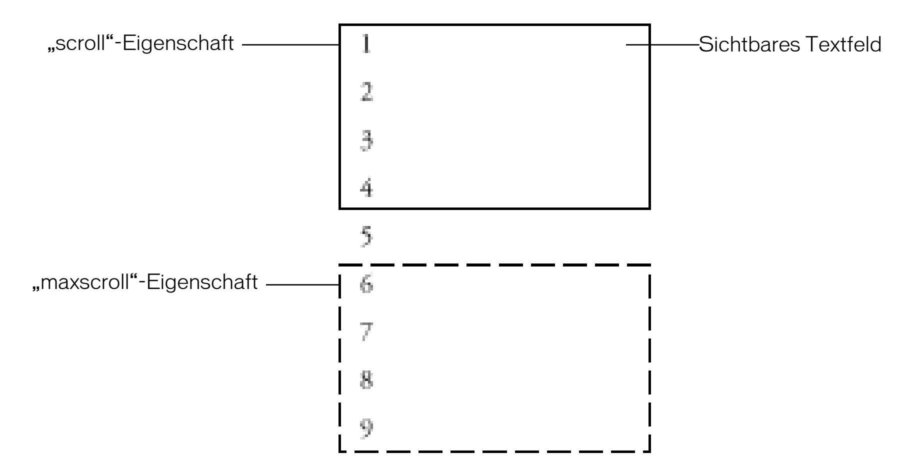
Zum Erzeugen eines Textfeldes mit Bildlauf können Sie die „scroll“- und „maxscroll“-Eigenschaften verwenden.



Sie können jedem Textfeld vom Typ „Eingabetext“ oder „Dynamischer Text“ im Bedienfeld „Textoptionen“ eine Variable zuordnen. Das Textfeld wirkt als Fenster, das den Wert dieser Variablen anzeigt.

Eine mit einem Textfeld verbundene Variable weist die `scroll`- und `maxscroll`-Eigenschaften auf. Diese Eigenschaften dienen zum Durchführen eines Bildlaufs bei Text in einem Textfeld. Mit der `scroll`-Eigenschaft wird die Nummer der obersten sichtbaren Zeile eines Textfeldes zurückgegeben. Sie können den Wert der Eigenschaft festlegen und abfragen. Mit der `maxscroll`-Eigenschaft wird die oberste sichtbare Zeile eines Textfeldes zurückgegeben, in dem die unterste Textzeile sichtbar ist. Sie können den Wert der Eigenschaft abfragen, aber nicht festlegen.

Angenommen, es liegt ein vier Zeilen langes Textfeld vor. Wenn es die Variable `speech` enthält, die neun Zeilen des Textfeldes füllen würde, kann jeweils nur ein Teil der Variablen `speech` angezeigt werden (mit einem durchgezogenen Kästchen gekennzeichnet):



Mit Punkt-Syntax wird folgendermaßen auf diese Eigenschaften zugegriffen:

```
textFieldVariable.scroll
myMovieClip.textFieldVariable.scroll
textFieldVariable.maxscroll
myMovieClip.textFieldVariable.maxscroll
```

So erstellen Sie ein Textfeld mit Bildlauf:

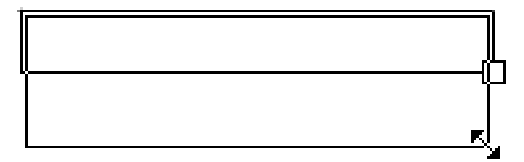
1 Ziehen Sie ein Textfeld auf die Bühne.

2 Wählen Sie „Fenster“ > „Bedienfelder“ > „Textoptionen“.

3 Wählen Sie im Popup-Menü die Option „Eingabetext“.

4 Geben Sie im Feld „Variable“ den Variablennamen **text** ein.

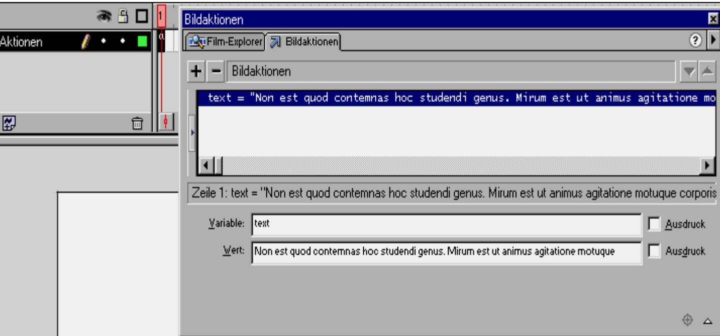
5 Ziehen Sie die rechte untere Ecke des Feldes, um die Größe des Textfeldes festzulegen.



6 Wählen Sie „Fenster“ > „Aktionen“.

7 Wählen Sie in der Hauptzeitleiste Bild 1 aus, und ordnen Sie eine Aktion `set variable` zu, bei der der Wert von `text` gesetzt wird.

Solange der Variablen kein Wert zugewiesen ist, wird im Feld kein Text angezeigt. Obwohl Sie diese Aktion jedem Bild, jeder Schaltfläche oder Filmsequenz zuordnen können, ist es daher zu empfehlen, sie Bild 1 der Hauptzeitleiste zuzuordnen, wie in diesem Beispiel:



8 Wählen Sie „Fenster“ > „Allgemeine Bibliotheken“ > „Schaltflächen“, und ziehen Sie eine Schaltfläche auf die Bühne.

9 Drücken Sie Alt (Windows) oder Wahltaste (Macintosh), und ziehen Sie die Schaltfläche, um eine Kopie zu erstellen.

10 Wählen Sie die oberste Schaltfläche aus, und wählen Sie „Fenster“ > „Aktionen“.

11 Ziehen Sie die Aktion `Variable einstellen` aus der Werkzeugleiste in das Skriptfenster des Bedienfeldes „Aktionen“.

12 Geben Sie im Feld „Variable“ den Wert **text.scroll** ein.

13 Geben Sie im Feld „Wert“ den Wert **text.scroll -1** ein, und aktivieren Sie das Kontrollkästchen „Ausdruck“.

14 Wählen Sie die NACH-UNTEN-TASTE aus, und weisen Sie die folgende Aktion für `Variable einstellen` zu:

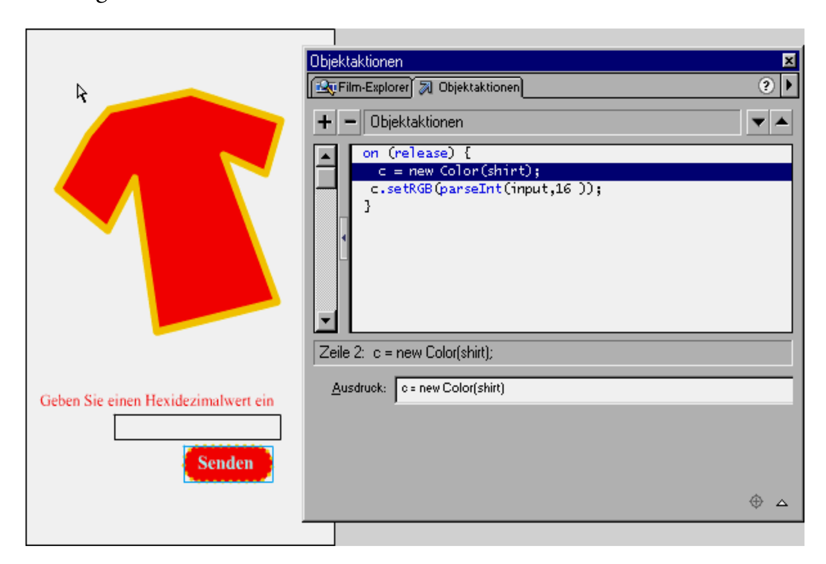
```
text.scroll = text.scroll+1;
```

15 Wählen Sie „Steuerung“ > „Film testen“, um das Textfeld mit Bildlauf zu testen.

Weitere Informationen zu den `scroll`- und `maxscroll`-Eigenschaften finden Sie in Kapitel 7, „Referenz zu ActionScript“.

Setzen von Farbwerten

Sie können die Methoden des vordefinierten Color-Objektes einsetzen, um die Farben einer Filmsequenz anzupassen. Mit der Methode `setRGB` werden dem Objekt hexadezimale RGB-Werte (Rot, Grün, Blau) zugewiesen, und mit der Methode `setTransform` werden für jede der Farbkomponenten Rot, Grün, Blau und Transparenz (Alpha) ein Prozentsatz und ein Offset-Wert festgelegt. Im folgenden Beispiel wird die Farbe eines Objektes nach einer Benutzereingabe mit `setRGB` geändert.



Die Schaltflächenaktion erzeugt ein Objekt vom Typ „Color“ und ändert die Farbe des T-Shirts entsprechend der Benutzereingabe.

Wenn Sie das Color-Objekt verwenden möchten, müssen Sie eine Instanz des Objektes erzeugen und diese in eine Filmsequenz aufnehmen.

So setzen Sie den Farbwert einer Filmsequenz:

1 Wählen Sie die Filmsequenzinstanz auf der Bühne aus, und wählen Sie „Fenster“ > „Bedienfelder“ > „Instanz“.

2 Geben Sie im Feld „Name“ den Instanznamen **colorTarget** ein.

3 Ziehen Sie ein Textfeld auf die Bühne.

4 Wählen Sie „Fenster“ > „Bedienfelder“ > „Textoptionen“, und weisen Sie den Variablennamen **input** zu.

5 Ziehen Sie eine Schaltfläche auf die Bühne, und wählen Sie sie aus.

6 Wählen Sie „Fenster“ > „Aktionen“.

7 Ziehen Sie die Aktion `set variable` aus der Werkzeugleiste in das Skriptfenster.

8 Geben Sie im Feld „Variable“ den Wert **c** ein.

9 Wählen Sie in der Werkzeugleiste erst „Objekte“ und dann „Farben“ aus, und ziehen Sie `neue Farbe` in das Feld „Wert“.

10 Aktivieren Sie das Kontrollkästchen „Ausdruck“.

11 Klicken Sie auf die Schaltfläche „Zielpfad“, und wählen Sie `colorTarget` aus. Klicken Sie auf „OK“.

Dabei entsteht im Skriptfenster folgender Code:

```
on(release) {
    c = new Color(colorTarget);
}
```

12 Ziehen Sie die Aktion `evaluate` aus der Werkzeugleiste in das Skriptfenster.

13 Geben Sie im Feld „Ausdruck“ den Wert **c** ein.

14 Wählen Sie in der Kategorie „Objekte“ der Werkzeugleiste „Farben“ aus, und ziehen Sie `setRGB` in das Feld „Ausdruck“.

15 Wählen Sie „Funktionen“ aus, und ziehen Sie `parseInt` in das Feld „Ausdruck“.

Dabei entsteht folgender Code:

```
on(release) {
    c = new Color(colorTarget);
    c.setRGB(parseInt(string, radix));
}
```

16 Geben Sie als Argument „string“ für `parseInt` den Wert **input** ein.

Die zu parsende Zeichenfolge ist der Wert, der in das editierbare Textfeld eingegeben wird.

17 Geben Sie als Argument „radix“ für `parseInt` den Wert **16** ein.

Hierbei handelt es sich um die Basis des Zahlensystems, nach dem geparst wird. In diesem Fall ist 16 die Basis des vom Color-Objekt verwendeten Hexadezimalsystems. Dabei entsteht folgender Code:

```
on(release) {
    c = new Color(colorTarget);
    c.setRGB(parseInt(input, 16));
}
```

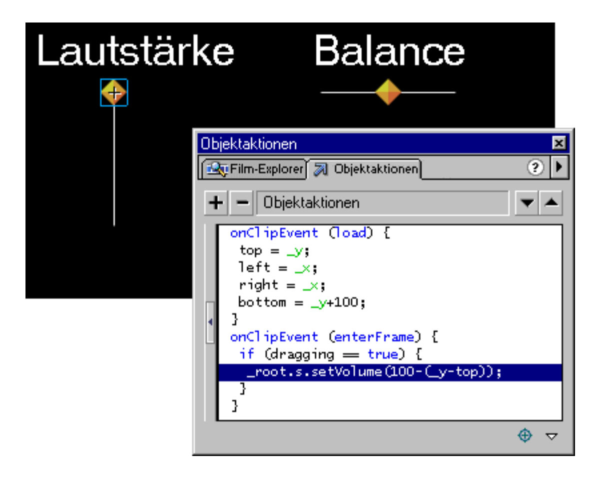
18 Wählen Sie „Steuerung“ > „Film testen“, um die Farbe der Filmsequenz zu ändern.

Weitere Informationen zu den Methoden des Color-Objektes finden Sie in den entsprechenden Einträgen in Kapitel 7, „Referenz zu ActionScript“.

Erstellen von Sound-Steuerelementen

Mit dem vordefinierten Sound-Objekt wird Sound innerhalb eines Filmes gesteuert. Um die Methoden des Sound-Objektes anwenden zu können, müssen Sie zuerst ein neues Objekt vom Typ „Sound“ erzeugen. Mit der Methode `attachSound` kann dann ein Sound aus der Bibliothek in einen Film eingefügt werden, während der Film abgespielt wird.

Die Methode `setVolume` des Sound-Objektes steuert die Lautstärke, und die Methode `setPan` passt die Balance (links/rechts) eines Sounds an. Im folgenden Beispiel werden mit `setVolume` und `setPan` vom Benutzer einstellbare Steuerelemente für Lautstärke und Balance erzeugt.



Wenn der Benutzer den Lautstärke-Schieberegler betätigt, wird die Methode `setVolume` *aufgerufen.*

So ordnen Sie einer Zeitleiste einen Sound zu:

1 Wählen Sie „Datei“ > „Importieren“, um einen Sound zu importieren.

2 Wählen Sie den Sound aus der Bibliothek aus, und wählen Sie im Menü „Optionen“ die Option „Verknüpfung“.

3 Wählen Sie „Dieses Symbol exportieren“ aus, und weisen Sie den Bezeichner **mySound** zu.

4 Wählen Sie in der Hauptzeitleiste Bild 1 aus, und wählen Sie „Fenster“ > „Aktionen“.

5 Ziehen Sie die Aktion `set variable` aus der Werkzeugleiste in das Skriptfenster.

6 Geben Sie im Feld „Wert“ den Wert **s** ein.

7 Wählen Sie in der Werkzeugliste erst „Objekte“ und dann „Sound“ aus, und ziehen Sie `neuer Sound` in das Feld „Wert“.

   Dabei entsteht folgender Code:

```
s = new Sound();
```

8 Doppelklicken Sie in der Werkzeugleiste auf die Aktion `evaluate`.

9 Geben Sie im Feld „Ausdruck“ den Wert **s** ein.

10 Wählen Sie in der Kategorie „Objekte“ der Werkzeugliste den Eintrag „Sound“ aus, und ziehen Sie `attachSound` in das Feld „Ausdruck“.

11 Geben Sie als Argument „ID“ von `attachSound` den Wert **„mySound“** ein.

12 Doppelklicken Sie in der Werkzeugleiste auf die Aktion `evaluate`.

13 Geben Sie im Feld „Ausdruck“ den Wert **s** ein.

14 Wählen Sie in der Kategorie „Objekte“ den Eintrag „Sound“ aus, und ziehen Sie `start` in das Feld „Ausdruck“.

   Dabei entsteht folgender Code:

   s = new Sound();

```
s.attachSound("mySound");
s.start();
```

15 Wählen Sie „Steuerung“ > „Film testen“, um den Sound anzuhören.

So erzeugen Sie einen Lautstärke-Schieberegler:

1 Ziehen Sie eine Schaltfläche auf die Bühne.

2 Wählen Sie die Schaltfläche aus, und wählen Sie „Einfügen“ > „In Symbol konvertieren“. Wählen Sie das Verhalten der Filmsequenz.

   Damit wird eine Filmsequenz mit der Schaltfläche im ersten Bild erzeugt.

3 Wählen Sie die Filmsequenz aus, und wählen Sie „Bearbeiten“ > „Symbole bearbeiten“.

4 Wählen Sie die Schaltfläche aus, und wählen Sie „Fenster“ > „Aktionen“.

5 Geben Sie die folgenden Aktionen ein:

```
on (press) {
    startDrag ("", false, left, top, right, bottom);
    dragging = true;
}
on (release, releaseOutside) {
    stopDrag ();
    dragging = false;
}
```

   Die Parameter `left`, `top`, `right` und `bottom` von `startDrag` sind Variablen, die in einer Filmsequenzaktion gesetzt werden.

6 Wählen Sie „Bearbeiten“ > „Film bearbeiten“, um zur Hauptzeitleiste zurückzukehren.

7 Wählen Sie die Filmsequenz auf der Bühne aus.

8 Geben Sie die folgenden Aktionen ein:

```
onClipEvent (load) {
    top=_y;
    left=_x;
    right=_x;
    bottom=_y+100;
}

onClipEvent(enterFrame){
    if (dragging==true){
       _root.s.setVolume(100-(_y-top));
    }
}
```

9 Wählen Sie „Steuerung“ > „Film testen“, um den Lautstärke-Schieberegler zu verwenden.

So erstellen Sie einen Balance-Schieberegler:

1 Ziehen Sie eine Schaltfläche auf die Bühne.

2 Wählen Sie die Schaltfläche aus, und wählen Sie „Einfügen“ > „In Symbol konvertieren“. Wählen Sie die Eigenschaften der Filmsequenz.

3 Wählen Sie die Filmsequenz aus, und wählen Sie „Bearbeiten“ > „Symbole bearbeiten“.

4 Wählen Sie die Schaltfläche aus, und wählen Sie „Fenster“ > „Aktionen“.

5 Geben Sie die folgenden Aktionen ein:

```
on (press) {
    startDrag ("", false, left, top, right, bottom);
    dragging = true;
}
on (release, releaseOutside) {
    stopDrag ();
    dragging = false;
}
```

Die Parameter `left`, `top`, `right` und `bottom` von `startDrag` sind Variablen, die in einer Filmsequenzaktion gesetzt werden.

6 Wählen Sie „Bearbeiten“ > „Film bearbeiten“, um zur Hauptzeitleiste zurückzukehren.

7 Wählen Sie die Filmsequenz auf der Bühne aus.

8 Geben Sie die folgenden Aktionen ein:

```
onClipEvent(load){
    top=_y;
    bottom=_y;
    left=_x-50;
    right=_x+50;
    center=_x;
}

onClipEvent(enterFrame){
    if (dragging==true){
       _root.s.setPan((_x-center)*2);
    }
}
```

9 Wählen Sie „Steuerung“ > „Film testen“, um den Balance-Schieberegler zu verwenden.

Weitere Informationen zu den Methoden des Sound-Objektes finden Sie in den entsprechenden Einträgen in Kapitel 7, „Referenz zu ActionScript“.

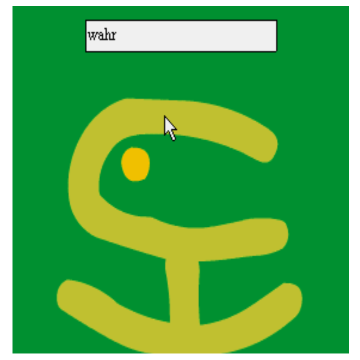
Erkennen von Kollisionen

Mit der Methode `hitTest` des MovieClip-Objektes können Kollisionen innerhalb eines Filmes erkannt werden. Mit der Methode `hitTest` wird überprüft, ob ein Objekt mit einer Filmsequenz kollidiert ist; es wird ein Boolescher Wert (`true` oder `false`) zurückgegeben. Sie können die Parameter der Methode `hitTest` zur Angabe der *x*- und *y*-Koordinaten eines Kollisionsbereiches verwenden, oder Sie können den Zielpfad einer anderen Filmsequenz als Kollisionsbereich angeben.

Jede Filmsequenz innerhalb eines Filmes ist eine Instanz des MovieClip-Objektes. Dadurch ist es möglich, wie im folgenden Beispiel Methoden des Objektes von einer beliebigen Instanz aus aufzurufen:

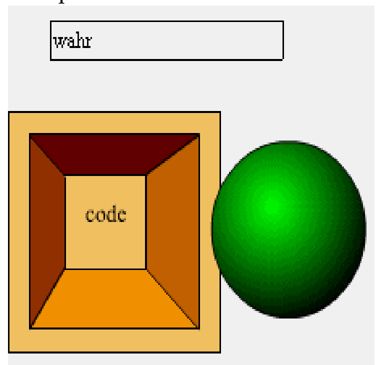
```
myMovieClip.hitTest(target);
```

Mit der Methode `hitTest` können Sie die Kollision einer Filmsequenz mit einem einzelnen Punkt testen.



Die Ergebnisse von `hitTest` *werden im Textfeld zurückgegeben.*

Mit der Methode `hitTest` können Sie auch auf eine Kollision von zwei Filmsequenzen testen.



Die Ergebnisse von `hitTest` *werden im Textfeld zurückgegeben.*

So führen Sie eine Kollisionserkennung zwischen einer Filmsequenz und einem Punkt auf der Bühne durch:

1 Wählen Sie eine Filmsequenz auf der Bühne aus.

2 Wählen Sie „Fenster“ > „Aktionen“, und öffnen Sie das Bedienfeld „Objektaktionen“.

3 Doppelklicken Sie in der Kategorie „Aktionen“ der Werkzeugleiste auf `trace`.

4 Aktivieren Sie das Kontrollkästchen „Ausdruck“, und geben Sie im Feld „Ausdruck“ Folgendes ein:

```
trace (this.hitTest(_root._xmouse, _root._ymouse, true);
```

   In diesem Beispiel werden die `_xmouse`- und `_ymouse`-Eigenschaften als *x*- und *y*-Koordinaten des Kollisionsbereiches verwendet. Die Ergebnisse werden im Modus „Film testen“ an das Ausgabefenster übergeben. Sie können die Ergebnisse auch in einem Textfeld auf der Bühne anzeigen lassen, und Sie können die Ergebnisse in einer `if` -Anweisung auswerten.

5 Wählen Sie „Steuerung“ > „Film testen“, und bewegen Sie die Maus über die Filmsequenz, um die Kollision zu testen.

So führen Sie eine Kollisionserkennung zwischen zwei Filmsequenzen durch:

1 Ziehen Sie zwei Filmsequenzen auf die Bühne, und geben Sie ihnen die Instanznamen **mcHitArea** und **mcDrag**.

2 Erzeugen Sie auf der Bühne ein Textfeld, und geben Sie im Feld „Variable“ von „Textoptionen“ den Wert **status** ein.

3 Wählen Sie `mcHitArea` aus, und wählen Sie „Fenster“ > „Aktionen“.

4 Doppelklicken Sie in der Werkzeugleiste auf `evaluate`.

5 Geben Sie mittels Auswahl aus der Werkzeugleiste im Feld „Ausdruck“ den folgenden Code ein:

```
_root.status=this.hitTest(_root.mcDrag);
```

6 Wählen Sie im Skriptfenster die Aktion `onClipEvent` aus, und wählen Sie das Ereignis `enterFrame`.

7 Wählen Sie `mcDrag` aus, und wählen Sie „Fenster“ > „Aktionen“.

8 Doppelklicken Sie in der Werkzeugleiste auf `startDrag`.

9 Aktivieren Sie das Kontrollkästchen „Maus zentrieren“.

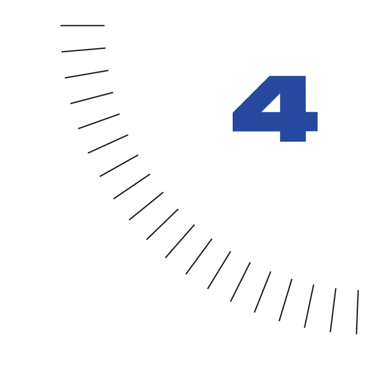
10 Wählen Sie im Skriptfenster die Aktion `onClipEvent` aus, und wählen Sie das Ereignis `Maus drücken`.

11 Doppelklicken Sie in der Werkzeugleiste auf `stopDrag`.

12 Wählen Sie im Skriptfenster die Aktion `onClipEvent` aus, und wählen Sie das Ereignis `Maus loslassen`.

13 Wählen Sie „Steuerung“ > „Film testen“, und ziehen Sie die Filmsequenz mit der Maus, um die Kollisionserkennung zu testen.

Weitere Informationen zur Methode `hitTest` finden Sie in Kapitel 7, „Referenz zu ActionScript“.



KAPITEL 4
Arbeiten mit Filmsequenzen

Bei einer Filmsequenz handelt es sich um einen kurzen Flash Film: er verfügt über eine eigene Zeitleiste und eigene Eigenschaften. Ein Filmsequenzsymbol in der Bibliothek kann mehrmals in einem Flash Film verwendet werden. Diese wiederholten Sequenzen werden als *Instanzen* der Filmsequenz bezeichnet. Filmsequenzen können ineinander verschachtelt werden. Um zwischen einzelnen Instanzen zu unterscheiden, können Sie jeder Instanz einen Instanznamen zuweisen.

Auf der Zeitleiste einer Filmsequenz können beliebige Objekte (sogar weitere Filmsequenzen) abgelegt werden. Filme, die über `loadMovie` in den Flash Player geladen wurden, werden ebenfalls als kurze Flash Filme behandelt. Bei jeder Filmsequenz, jedem geladenen Film und der Hauptzeitleiste eines Flash Filmes handelt es sich um ein Objekt mit Eigenschaften und Methoden, die mit ActionScript gesteuert werden können, um komplexe, nichtlineare Animationen und starke Interaktivität herzustellen.

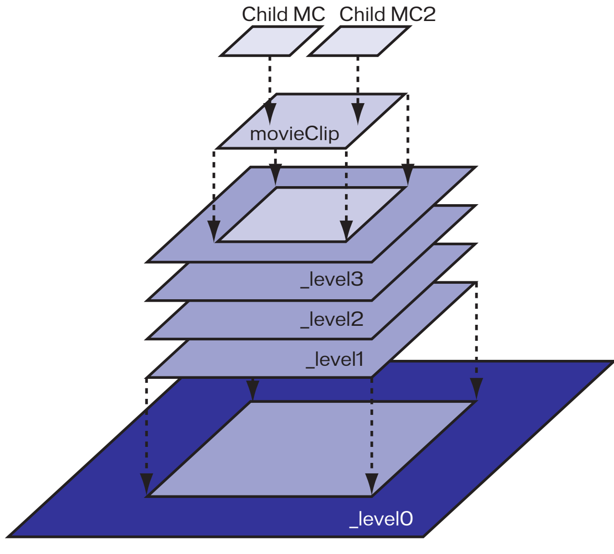
Sie steuern Filmsequenzen mit Hilfe von Aktionen und den Methoden des MovieClip-Objektes. Aktionen und Methoden können Bildern und Schaltflächen in einer Filmsequenz (als Bild- und Schaltflächenaktionen) oder einer bestimmten Instanz der Filmsequenz (als Filmsequenzaktionen) zugeordnet werden. Über die Aktionen in einer Filmsequenz können alle in einem Film vorhandenen Zeitleisten gesteuert werden. Hierbei muss die Zeitleiste durch einen entsprechenden Zielpfad ausgewiesen werden. Der Zielpfad gibt die Position der Zeitleiste im Film an.

Sie können eine Filmsequenz auch in eine Smart-Filmsequenz umwandeln. Eine solche Sequenz enthält ActionScript-Elemente und kann ohne das Bedienfeld „Aktionen“ neu programmiert werden. Dadurch erleichtern Smart-Filmsequenzen die Weitergabe von Objekten mit komplexer ActionScript-Logik vom Programmierer an den Designer.

Erläuterungen zur Verwendung mehrerer Zeitleisten

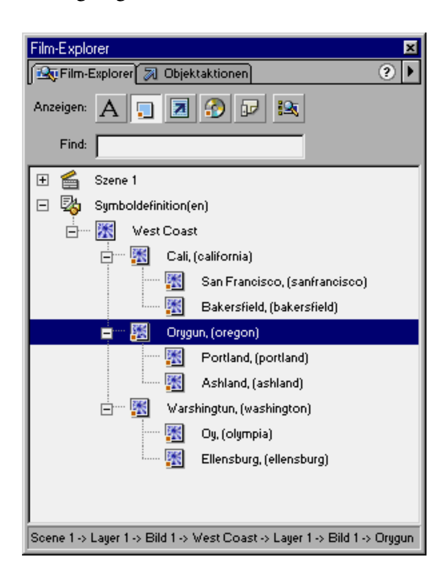
Jeder Flash Film verfügt über eine Hauptzeitleiste auf Ebene 0 im Flash Player. Mit der Aktion `loadMovie` können Sie weitere Flash Filme (SWF-Dateien) in eine beliebige Ebene über Ebene 0 in den Flash Player laden (z. B. in Ebene 1, Ebene 2 oder Ebene 15). Jeder in eine Ebene im Flash Player geladene Film weist eine Zeitleiste auf.

Unabhängig von der Ebene des Filmes können auf der Zeitleiste eines Flash Filmes Filmsequenzinstanzen eingefügt werden. Diese Filmsequenzinstanzen weisen wiederum Zeitleisten auf und können weitere Filmsequenzen mit eigenen Zeitleisten enthalten. Die Zeitleisten der Filmsequenzen und Ebenen im Flash Player folgen einer bestimmten Hierarchie und ermöglichen damit das Organisieren und einfache Steuern der Objekte in Ihrem Film.



Die Hierarchie der Ebenen und Filmsequenzen im Flash Player.

In Flash wird diese Hierarchie der Ebenen und Filmsequenzen als *Anzeigeliste* bezeichnet. Während der Entwicklung in Flash können Sie die Anzeigeliste im Film-Explorer aufrufen. Wenn Sie den Film im Filmtestmodus, im Flash Player oder in einem Webbrowser abspielen, steht die Anzeigeliste im Debugger zur Verfügung.



Der Film-Explorer zeigt die Hierarchie der Zeitleisten, die als Anzeigeliste bezeichnet wird.

Die Zeitleisten in einem Flash Film sind Objekte und weisen die Merkmale (Eigenschaften) und Fähigkeiten (Methoden) des vordefinierten MovieClip-Objektes auf. Zwischen den Zeitleisten bestehen bestimmte Beziehungen, die von ihrer jeweiligen Position in der Anzeigeliste abhängen. Wenn Sie Änderungen an einer Zeitleiste vornehmen, in die weitere Zeitleisten eingebunden sind, beeinflussen diese Änderungen auch die untergeordneten Zeitleisten. Wenn `portland` beispielsweise ein Unterelement von `oregon` ist und Sie die Eigenschaft `_xscale` für `oregon` ändern, wird gleichzeitig `portland` neu skaliert.

Die Zeitleisten können Meldungen untereinander austauschen. So kann z. B. über eine Aktion auf dem letzten Bild einer Filmsequenz eine neue Filmsequenz gestartet werden.

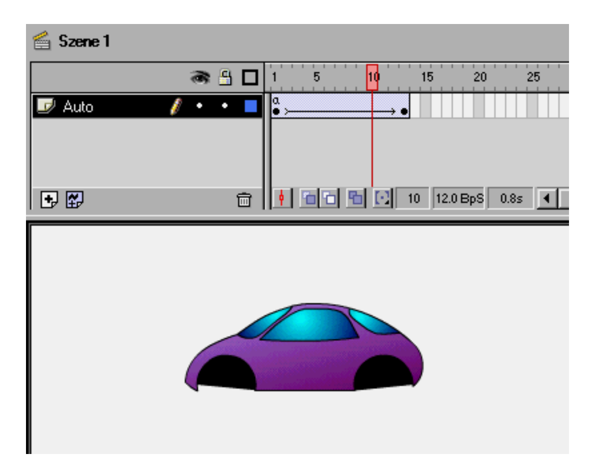
Erläuterungen zu den hierarchischen Beziehungen zwischen Zeitleisten

Wenn Sie eine Filmsequenzinstanz auf die Zeitleiste einer anderen Filmsequenz setzen, enthält das eine Filmsequenzsymbol die Instanz der anderen Filmsequenz: die erste Filmsequenz ist das *untergeordnete* und die zweite Filmsequenz das *übergeordnete* Element. Die Hauptzeitleiste eines Flash Filmes ist das übergeordnete Element aller Filmsequenzen auf dieser Ebene.

Die Beziehungen zwischen übergeordneten und untergeordneten Filmsequenzen folgen einem hierarchischen Schema. Führen Sie sich in diesem Zusammenhang die Hierarchie eines Computers vor Augen: die Festplatte verfügt über ein Stammverzeichnis (bzw. Stammordner) und Unterverzeichnisse. Das Stammverzeichnis entspricht hierbei der Hauptzeitleiste eines Flash Films: es dient als übergeordnetes Element für alle übrigen Elemente. Die Unterverzeichnisse entsprechen den Filmsequenzen. Sie können Unterverzeichnisse anlegen, um verwandte Inhalte zu gruppieren.

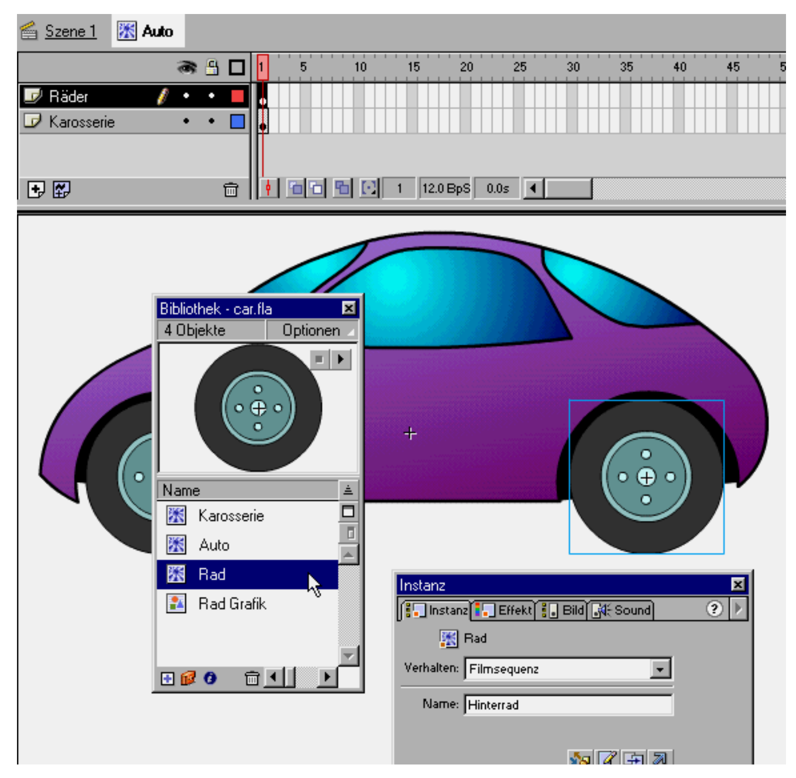
In ähnlicher Weise dient die Filmsequenzhierarchie in Flash zum Organisieren von verwandten visuellen Objekten, wobei sich der Aufbau oftmals am realen Verhalten der Objekte orientiert. Jede Änderung in einer übergeordneten Filmsequenz wird für die untergeordneten Sequenzen übernommen.

Angenommen, Sie möchten einen Flash Film erstellen, in dem sich ein Auto über die Bühne bewegt. Sie können z. B. ein Filmsequenzsymbol für das Auto verwenden und einen Bewegungs-Tween festlegen, um das Auto über die Bühne zu bewegen.



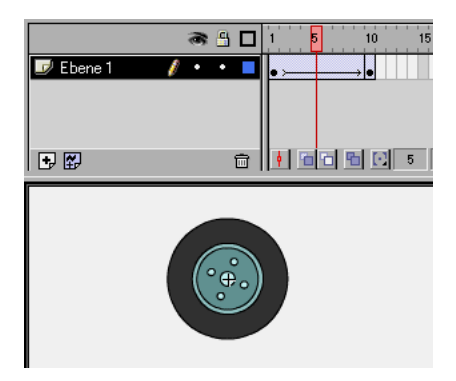
Ein Bewegungs-Tween bewegt die Auto-Filmsequenz auf der Hauptzeitleiste.

Das Auto wird von der Seite gezeigt, und zwei Räder sind sichtbar. Nachdem Sie die Bewegung des Autos festgelegt haben, möchten Sie drehende Räder hinzufügen. Hierzu legen Sie eine Filmsequenz für das Rad fest und erstellen zwei Instanzen dieser Filmsequenz mit den Namen `frontWheel` und `backWheel`. Anschließend positionieren Sie die Räder auf der Zeitleiste der Auto-Filmsequenz, nicht auf der Hauptzeitleiste. Als Unterelemente von `car` folgen `frontWheel` und `backWheel` allen Änderungen, die für `car` vorgenommen werden. Das bedeutet, dass sie sich zusammen mit dem Auto über die Bühne bewegen, wenn dessen Tween abgearbeitet wird.



Die Instanzen `frontWheel` und `backWheel` befinden sich auf der Zeitleiste der Filmsequenz `car`.

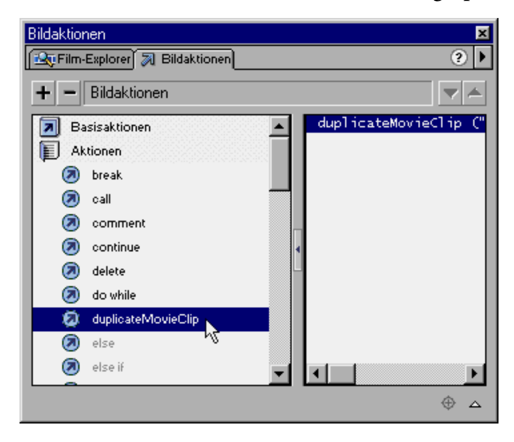
Für die Drehung der Räder können Sie einen Bewegungs-Tween festlegen, der das Rad-Symbol rotiert und so beide Instanzen animiert. Auch wenn Sie Änderungen an `frontWheel` und `backWheel` vornehmen, werden diese weiterhin durch den Bewegungs-Tween der übergeordneten Filmsequenz beeinflusst. Die Räder drehen sich und bewegen sich zusätzlich mit der übergeordneten Filmsequenz `car` über die Bühne.



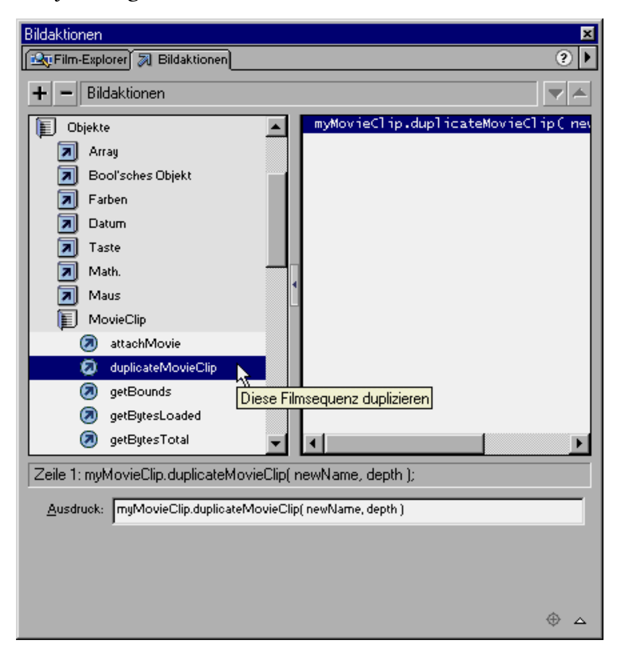
Senden von Meldungen zwischen Zeitleisten

Sie können Meldungen von einer Zeitleiste an eine andere senden. Die eine Zeitleiste beinhaltet hierbei die Aktion und wird als *Abspielsteuerung* bezeichnet, und die andere empfängt diese Aktion als *Ziel*. Sie können den Bildern oder Schaltflächen in einer Zeitleiste Aktionen zuordnen, oder die Aktion mit der Filmsequenz selbst verbinden, wenn es sich bei der Zeitleiste um eine Filmsequenz handelt.

Verwenden Sie Aktionen aus der Kategorie „Aktionen“ oder die Methoden des MovieClip-Objektes aus der Kategorie „Objekte“ im Bedienfeld „Aktionen“, um die gewünschte Zeitleiste als Ziel festzulegen. Sie können z. B. mit der Aktion `duplicateMovieClip` Filmsequenzinstanzen als Ziel festlegen und Kopien dieser Instanzen erstellen, während ein Film abgespielt wird.



Sie können Aktionen aus der Kategorie „Aktionen“ verwenden, um eine Zeitleiste als Ziel festzulegen.



Sie können Methoden des MovieClip-Objektes verwenden, um eine Zeitleiste als Ziel festzulegen.

Wenn Sie mehrere Aktionen für dasselbe Ziel ausführen möchten, können Sie die Aktion `with` verwenden. Ähnlich wie bei der JavaScript-Anweisung `with` müssen Sie bei der ActionScript-Aktion `with` die gewünschte Zeitleiste nur einmal als Ziel festlegen. Anschließend kann eine Abfolge mehrerer Aktionen für die Sequenz ausgeführt werden, ohne dass die Zeitleiste jedes Mal erneut als Ziel festgelegt werden muss.

Auch mit der Aktion `tellTarget` können mehrere Aktionen für dasselbe Ziel ausgeführt werden.

Für die Kommunikation zwischen Zeitleisten sind die folgenden Schritte erforderlich:

- Geben Sie einen Instanznamen für die Zielfilmsequenz ein.

  Öffnen Sie das Bedienfeld „Instanz" („Fenster" > „Bedienfelder" > „Instanz"), um den Namen für eine Filmsequenzinstanz festzulegen. Wenn Zeitleisten in verschiedene Ebenen geladen wurden, dient die Nummer der jeweiligen Ebene als Instanzname, z. B. `_level6`.

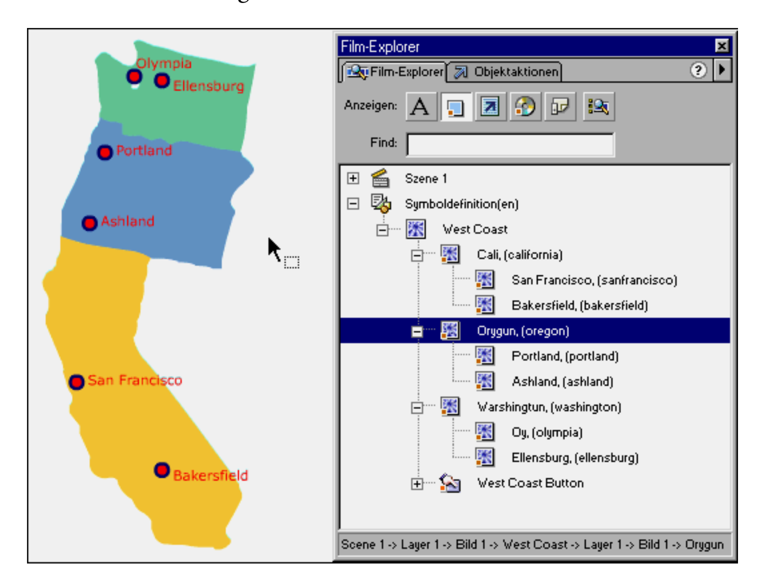
- Geben Sie im Bedienfeld „Aktionen" den Zielpfad zum Instanznamen ein.

  Sie können den Pfad manuell eingeben oder die Zielfilmsequenz im Dialogfeld „Zielpfad einfügen" auswählen. (Siehe „Angeben von Zielpfaden" auf Seite 123.)

Anmerkung: Wenn eine Zeitleiste einer Filmsequenz während der Wiedergabe als Ziel festgelegt werden soll, muss sie sich auf der Bühne befinden.

Erläuterungen zu absoluten und relativen Zielpfaden

Der Zielpfad ist die Adresse der Zeitleiste, die als Ziel festgelegt werden soll. Die Anzeigeliste für Zeitleisten in Flash kann mit der Datei- und Ordnerhierarchie auf einem Webserver verglichen werden.



Im Erstellungsmodus können Sie die Anzeigeliste der Filmsequenzen im Film-Explorer aufrufen.

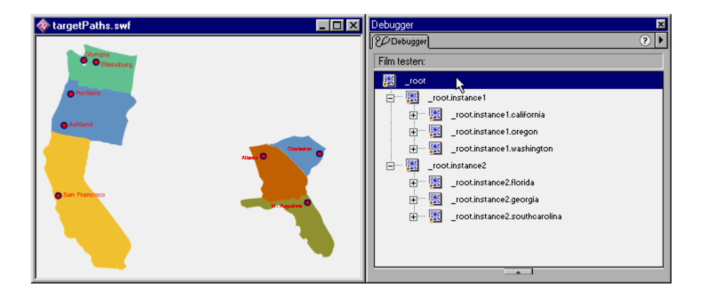
Wie auf einem Webserver kann der Verweis auf eine Zeitleiste in Flash auf zwei Arten erfolgen: mit einem absoluten oder einem relativen Pfad. Der absolute Pfad einer Instanz ist immer gleich, unabhängig davon, welche Zeitleiste die Aktion aufruft. Angenommen, der absolute Pfad für die Instanz `california` lautet `_level0.westCoast.california`. Er bleibt stets gleich. Ein relativer Pfad hingegen variiert abhängig von dem Ort, von dem aus der Aufruf erfolgt. Für `california` beispielsweise lautet der relative Pfad von `sanfrancisco` aus `_parent`, beim Aufruf von `portland` aus jedoch `_parent._parent.california`.

Anmerkung: Weitere Informationen über den Film-Explorer finden Sie im *Flash Benutzerhandbuch*.

Ein absoluter Pfad beginnt mit dem Namen der Ebene, in die der Film geladen wurde, und durchläuft die Anzeigeliste bis zur eigentlichen Zielinstanz.

Der erste Film, der im Flash Player geöffnet wird, wird in Ebene 0 geladen. Jedem weiteren Film muss eine Ebene (bzw. deren Nummer) zugewiesen werden. Der Zielname für eine Ebene lautet `_levelX` , wobei `X` die Nummer der Ebene darstellt, in die der Film geladen wird. So trägt der erste im Flash Player geöffnete Film z. B. die Bezeichnung `_level0` und ein in Ebene 3 geladener Film die Bezeichnung `_level3`.

Im folgenden Beispiel wurden zwei Filme in den Player geladen, `TargetPaths.swf` in Ebene 0 und `EastCoast.swf` in Ebene 5. Die Ebenen werden im Debugger angezeigt, wobei Ebene 0 als `_root` angegeben ist.



Der Debugger führt im Filmtestmodus die absoluten Pfade aller Zeitleisten in der Anzeigeliste auf.

Für eine Instanz gilt immer derselbe absolute Pfad, unabhängig davon, ob sie durch eine Aktion in einer Instanz auf derselben Ebene oder durch eine Aktion auf einer anderen Ebene aufgerufen wird. Der absolute Pfad in Punkt-Syntax für die Instanz `bakersfield` auf Ebene 0 lautet z. B.:

```
_level0.california.bakersfield
```

In der Schrägstrich-Syntax werden, wie im folgenden Beispiel, Schrägstriche anstelle der Punkte verwendet:

```
_level0/california/bakersfield
```

Wenn Sie zwischen Filmen auf unterschiedlichen Ebenen kommunizieren möchten, müssen Sie im Zielpfad den Namen der Ebene verwenden. So wird die Instanz `atlanta` in der Instanz `portland` z. B. wie folgt adressiert:

```
_level5.georgia.atlanta
```

In der Punkt-Syntax können Sie auch mit dem Alias `_root` auf die Hauptzeitleiste der aktuellen Ebene verweisen. Für die Hauptzeitleiste, oder `_level0`, verweist der Alias „_root" auf `_level0`, sofern der Verweis aus einer Sequenz erfolgt, die sich ebenfalls auf `_level0` befindet. Für einen in `_level5` geladenen Film entspricht `_root` dann `_level5`, wenn der Verweis aus einer Filmsequenz erfolgt, die sich ebenfalls auf Ebene 5 befindet. So kann eine in der Instanz `southcarolina` aufgerufene Aktion z. B. über den folgenden Pfad auf die Instanz `florida` verweisen:

```
_root.eastCoast.florida
```

In der Schrägstrich-Syntax können Sie das Zeichen `/` verwenden, um auf die Hauptzeitleiste der aktuellen Ebene zu verweisen, wie im folgenden Beispiel:

```
/eastCoast/florida
```

In der Punkt-Syntax können Sie sowohl bei absoluter als auch bei relativer Pfadangabe dieselben Regeln für die Zielpfadangabe verwenden, um eine Variable auf einer Zeitleiste oder eine Eigenschaft eines Objektes zu identifizieren. Die folgende Anweisung setzt z. B. die Variable „name" in der Instanz „form" auf den Wert `"Gilbert"`:

```
_root.form.name = "Gilbert";
```

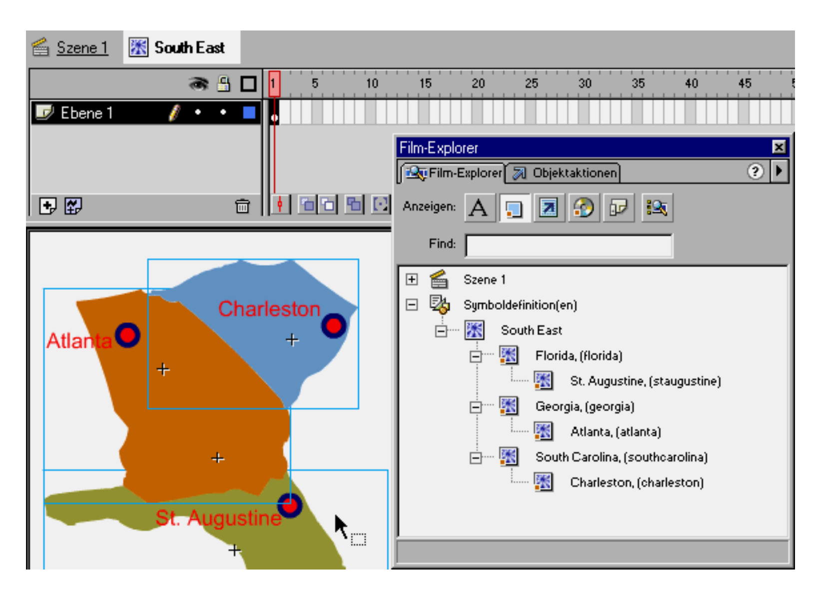
In Schrägstrich-Syntax können Sie eine Variable auf einer Zeitleiste sowohl bei absoluter als auch bei relativer Pfadangabe durch Voranstellen eines Doppelpunktes (`:`) vor dem Variablennamen identifizieren:

```
/form:name = "Gilbert";
```

Ein relativer Pfad hängt von der Beziehung zwischen der Zeitleiste, die die Abspielsteuerung darstellt, und der Zielzeitleiste ab. Sie können einen relativen Pfad angeben, wenn Sie Aktionen wiederverwenden möchten, weil eine Aktion abhängig vom Ort, von dem aus sie aufgerufen wird, auf unterschiedliche Zeitleisten verweisen kann. Relative Pfade können ausschließlich auf Ziele innerhalb ihrer jeweiligen Aufrufebene im Flash Player verweisen. Verweise auf Filme in anderen Ebenen sind nicht möglich. So können Sie in einer Aktion auf `_level0` z. B. keinen relativen Pfad angeben, der auf eine Zeitleiste in `_level5` verweist.

In der Punkt-Syntax ermöglicht das Schlüsselwort `this` in einem relativen Zielpfad einen Verweis auf die aktuelle Zeitleiste. Mit dem Alias `_parent` verweisen Sie in einem relativen Pfad auf die Zeitleiste, die der aktuellen Zeitleiste übergeordnet ist. Der Alias `_parent` kann mehrmals angegeben werden, um jeweils auf die nächste übergeordnete Ebene der Filmsequenzhierarchie innerhalb derselben Ebene im Flash Player zu gelangen. So wird mit `_parent._parent` z. B. eine Filmsequenz gesteuert, die sich in der Hierarchie zwei Ebenen über der Aufrufebene befindet.

Im folgenden Beispiel ist jede Stadt (`charleston`, `atlanta` und `staugustine`) ein untergeordnetes Element einer Staateninstanz und jeder Staat (`southcarolina`, `georgia` und `florida`) ein untergeordnetes Element der Instanz `eastCoast`.



Im Film-Explorer werden die Beziehungen zwischen übergeordneten und untergeordneten Filmsequenzen angezeigt.

In einer Aktion auf der Zeitleiste für die Instanz `charleston` kann der folgende Zielpfad für einen Verweis auf die Instanz `southcarolina` verwendet werden:

```
_parent
```

Um in einer Aktion in der Instanz `charleston` auf die Instanz `eastCoast` zu verweisen, können Sie den folgenden relativen Pfad verwenden:

```
_parent._parent
```

In Schrägstrich-Syntax geben Sie zwei Punkte ein (`..`), um auf die nächste übergeordnete Ebene in der Hierarchie zu gelangen. Für einen Verweis auf `eastCoast` in einer Aktion in `charleston` geben Sie z. B. den folgenden Pfad an:

```
../..
```

Wenn Sie in einer Aktion auf der Zeitleiste für `charleston` auf `atlanta` verweisen möchten, können Sie in der Punkt-Syntax den folgenden relativen Pfad verwenden:

```
_parent._parent.georgia.atlanta
```

Geben Sie relative Pfade an, wenn Sie ein Skript wiederverwenden möchten. Sie können einer Filmsequenz z.B. das folgende Skript hinzufügen, mit dem die Filmsequenz auf der nächsten übergeordneten Ebene um 150% vergrößert wird:

```
onClipEvent (load) {
   _parent._xscale = 150;
   _parent._yscale = 150;
}
```

Dieses Skript können Sie anschließend auf die Zeitleiste jeder beliebigen Filmsequenz setzen.

Weitere Informationen über Zielangaben (Adressierung) und die Punkt-Syntax finden Sie unter „Schreiben von Skripts mit ActionScript“ auf Seite 51.

Weitere Informationen zur Punkt- und Schrägstrich-Syntax finden Sie unter „Verwenden der ActionScript-Syntax“ auf Seite 51.

Angeben von Zielpfaden

Wenn Sie eine Filmsequenz oder einen geladenen Film steuern möchten, müssen Sie das gewünschte Ziel durch Angabe eines Zielpfads festlegen. Eine Filmsequenz muss einen Instanznamen aufweisen, damit diese als Ziel festgelegt werden kann. Sie können ein Ziel auf mehrere Arten festlegen:

- Geben Sie einen Zielpfad im Bedienfeld „Aktionen“ über die Schaltfläche und das Dialogfeld „Zielpfad einfügen“ ein.
- Geben Sie den Zielpfad für die Filmsequenz manuell in Ihr Skript ein.
- Erstellen Sie einen Ausdruck mit einem Verweis auf eine Filmsequenz oder mit den vordefinierten Funktionen `targetPath` und `eval`.

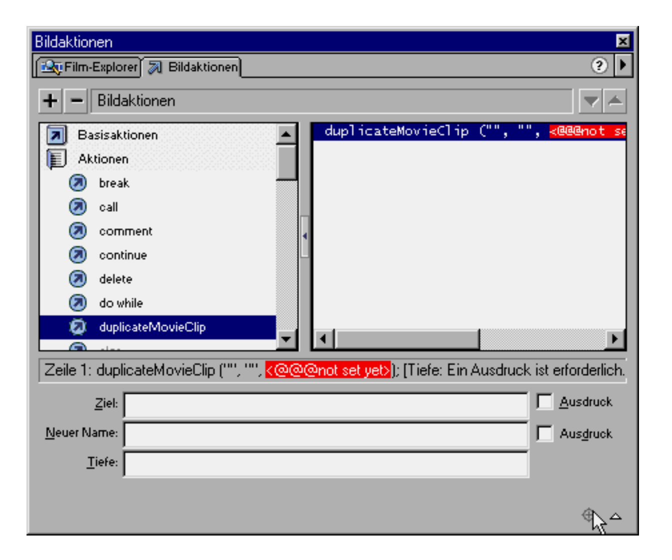
So fügen Sie einen Zielpfad über das Dialogfeld „Zielpfad einfügen“ ein:

1. Wählen Sie die Filmsequenz, das Bild oder die Schaltflächeninstanz aus, der Sie die Aktion zuordnen möchten.

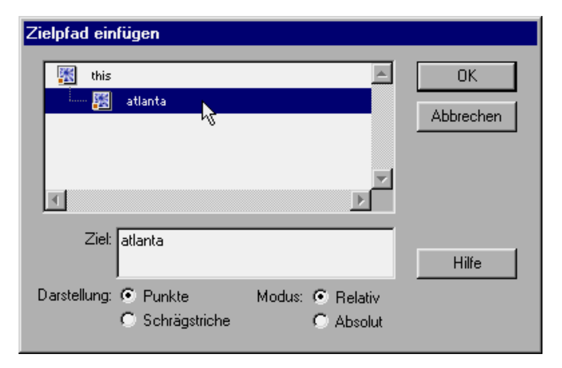
   Diese Zeitleiste wird die Abspielszeuerung.

2. Wählen Sie „Fenster“ > „Aktionen“ aus, um das Bedienfeld „Aktionen“ zu öffnen.
3. Wählen Sie in der Werkzeugliste aus der Kategorie „Aktionen“ eine Aktion oder in der Kategorie „Objekte“ aus der Kategorie „MovieClip“ eine Methode aus.
4. Klicken Sie auf das Zielfeld bzw. den Zielort im Skript, um den Zielpfad einzufügen.

5 Klicken Sie in der unteren rechten Ecke des Bedienfeldes „Aktionen“ auf die Schaltfläche „Zielpfad einfügen“, um das Dialogfeld „Zielpfad einfügen“ zu öffnen.



6 Wählen Sie im Dialogfeld „Zielpfad einfügen“ eine Syntax aus: Punkte (Voreinstellung) oder Schrägstriche.



7 Wählen Sie „Absolut“ oder „Relativ“ als Modus für den Zielpfad aus.

Siehe „Erläuterungen zu absoluten und relativen Zielpfaden“ auf Seite 119.

8 Geben Sie Ihr Ziel durch einen der folgenden Schritte an:

- Wählen Sie in der Anzeigeliste „Zielpfad einfügen“ eine Filmsequenz aus.
- Geben Sie im Feld „Ziel“ manuell einen absoluten oder einen relativen Pfad in Punkt-Syntax ein.

9 Klicken Sie auf „OK“.

So fügen Sie manuell einen Zielpfad ein:

Führen Sie die oben angegeben Schritte 1 bis 4 durch, und geben Sie im Bedienfeld „Aktionen“ einen absoluten oder relativen Zielpfad ein.

So verwenden Sie einen Ausdruck als Zielpfad:

1 Führen Sie die oben genannten Schritte 1 bis 4 durch.

2 Führen Sie einen der folgenden Schritte durch:

- Geben Sie manuell einen Verweis als Zielpfad ein. Wenn Sie einen Verweis eingeben, wird dieser zum Ermitteln des Zielpfads ausgewertet. Sie können einen Verweis als Parameter für die Aktion `with` verwenden. Im folgenden Beispiel wird die Variable `index` ausgewertet und mit 2 multipliziert. Das Ergebnis wird als Name der Filmsequenz innerhalb der Instanz `Block`, die wiedergegeben werden soll:

```
with (Board.Block[index*2]) {
    play();
}
```

- Wählen Sie in der Werkzeugliste aus der Kategorie „Funktionen“ die Funktion `targetPath` aus.

  Die Funktion `targetPath` wandelt einen Verweis auf eine Filmsequenz in eine Zeichenfolge um, die in Aktionen wie z. B. `tellTarget` verwendet werden kann.

  Im folgenden Beispiel wandelt die Funktion `targetPath` den Verweis `Board.Block[index*2+1]` in eine Zeichenfolge um:

```
tellTarget (targetPath (Board.Block[index*2+1])) {
    play();
}
```

  Das vorige Beispiel ist identisch mit dem folgenden Beispiel in Schrägstrich-Syntax:

```
tellTarget ("Board/Block:" + index*2+1)) {
    play();
}
```

- Wählen Sie in der Werkzeugliste aus der Kategorie „Funktionen“ die Funktion `eval` aus.

  Die Funktion `eval` wandelt eine Zeichenfolge in einen Verweis auf eine Filmsequenz um, der als Zielpfad in Aktionen wie `with` verwendet werden kann.

  Im folgenden Skript wird die Variable `i` ausgewertet, an die Zeichenfolge `"cat"` angehängt und das Ergebnis der Variable `x` zugewiesen. Die Variable „x“ stellt nun einen Verweis auf eine Filmsequenzinstanz dar und kann, wie im folgenden Beispiel, in einem Aufruf der Methoden des MovieClip-Objektes verwendet werden:

  ```
  x = eval ("cat" + i)
  x.play()
  ```

  Mit der Funktion `eval` kann auch direkt eine Methode aufgerufen werden:

  ```
  eval ("cat" + i).play()
  ```

Steuern von Zeitleisten durch Aktionen und Methoden

Sie können bestimmte Aktionen und Methoden des MovieClip-Objektes verwenden, um eine Filmsequenz oder eine geladene Ebene als Ziel festzulegen und Aktionen in dieser auszuführen. So wird mit der Aktion `setProperty` z.B. eine Eigenschaft einer Zeitleiste (wie `_width`) auf einen Wert (wie `100`) gesetzt. Die Funktionen einiger Methoden des MovieClip-Objektes entsprechen denen der Aktionen, mit denen Zeitleisten als Ziel festgelegt werden. Darüber hinaus sind zusätzliche Methoden vorhanden, z.B. `hitTest` und `swapDepths`. Unabhängig davon, ob Sie Aktionen oder Methoden verwenden, muss die entsprechende Zielzeitleiste im Flash Player geladen sein, wenn der Aufruf der Aktion oder Methode erfolgt.

In den folgenden Aktionen können Filmsequenzen als Ziel festgelegt werden: `loadMovie`, `unloadMovie`, `setProperty`, `startDrag`, `duplicateMovieClip` und `removeMovieClip`. Wenn Sie mit diesen Aktionen arbeiten, müssen Sie im Zielparameter der Aktion den Zielpfad zum Empfänger der Aktion eingeben. Bei einigen der Aktionen kann sowohl auf Filmsequenzen als auch auf Ebenen verwiesen werden. Bei anderen Aktionen können ausschließlich Filmsequenzen als Ziel festgelegt werden.

Die folgenden Methoden des MovieClip-Objektes steuern Filmsequenzen oder geladene Ebenen. Für diese Methoden sind keine entsprechenden Aktionen vorhanden: `attachMovie`, `getBounds`, `getBytesLoaded`, `getBytesTotal`, `globalToLocal`, `localToGlobal`, `hitTest` und `swapDepths`.

Wenn eine Aktion und eine Methode eine ähnliche Funktion bereitstellen, können Sie zwischen diesen beiden wählen. Entscheiden Sie nach Ihren persönlichen Vorlieben und danach, mit welchen Verfahren Sie beim Schreiben von Skripts in ActionScript eher vertraut sind.

Weitere Informationen über die Methoden des MovieClip-Objektes sowie Informationen zu den einzelnen Aktionen finden Sie in Kapitel 7, „Referenz zu ActionScript“ auf Seite 173.

Erläuterungen zu Methoden und Aktionen

Methoden werden aufgerufen, indem der Zielpfad zum Instanznamen gefolgt von einem Punkt, dem Namen der Methode und Argumenten angegeben werden, wie in den folgenden Anweisungen dargestellt:

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

In der ersten Anweisung wird mit der Methode `play` die Instanz `myMovieClip` abgespielt. In der zweiten Anweisung setzt die Methode `gotoAndPlay` den Abspielkopf in `childClip` (dies ist eine untergeordnete Instanz der Instanz `parentClip`) auf Bild 3 und startet die Wiedergabe.

Aktionen, mit denen Zeitleisten gesteuert werden, verfügen über einen Zielparameter, mit dem der Zielpfad angegeben wird. Im folgenden Skript wird z. B. in der Aktion `startDrag` die Instanz `customCursor` als Ziel festgelegt und zum Verschieben freigegeben:

```
on(press){
   startDrag("customCursor");
}
```

Beim Verwenden von Methoden rufen Sie die Methode am Ende des Zielpfads auf. In der folgenden Anweisung wird z. B. dieselbe `startDrag`-Funktion ausgeführt:

```
customCursor.startDrag();
```

Anweisungen, in denen Methoden des MovieClip-Objektes eingesetzt werden, sind in der Regel kürzer, weil die Aktion `tellTarget` nicht angegeben werden muss. Es wird empfohlen, die Aktion `tellTarget` nicht zu verwenden, weil sie nicht dem Standard ECMA-262 entspricht.

Wenn Sie z. B. die Filmsequenz `myMovieClip` mit Hilfe der Methoden des MovieClip-Objektes wiedergeben möchten, können Sie folgenden Code verwenden:

```
myMovieClip.play();
```

Der nachstehende Code führt zum selben Ergebnis. Hierbei wird jedoch die Aktion `tellTarget` verwendet.

```
tellTarget ("myMovieClip") {
   play();
}
```

Verwenden mehrerer Methoden oder Aktionen zum Festlegen einer Zeitleiste als Ziel

Bei der Aktion `with` müssen Sie eine Filmsequenz nur einmal als Ziel festlegen. Anschließen können mehrere Aktionen für diese Sequenz ausgeführt werden. Die Aktion `with` kann für alle ActionScript-Objekte verwendet werden (z. B. für Array, Color oder Sound), nicht nur für Filmsequenzen. Die Aktion `tellTarget` ähnelt der Aktion `with`. Allerdings wird nicht empfohlen, `tellTarget` zu verwenden, weil diese Aktion nicht für alle ActionScript-Objekte verwendet werden kann und nicht ECMA-262 entspricht.

Bei der Aktion `with` wird ein Objekt als Parameter übergeben. Das angegebene Objekt wird am Ende des aktuellen Zielpfads angefügt. Alle Aktionen, die sich innerhalb der Aktion `with` befinden, werden innerhalb des neuen Zielpfads, oder *Gültigkeitsbereichs*, ausgeführt. Im folgenden Skript auf der Hauptzeitleiste wird z. B. das Objekt `donut.hole` an die Aktion `with` übergeben, um anschließend die Eigenschaften von `hole` zu ändern:

```
with (donut.hole){
   _alpha = 20;
   _xscale = 150;
   _yscale = 150;
}
```

Durch Angabe der Aktion „with“ werden die Anweisungen innerhalb der Aktion so behandelt, als würden sie von der Zeitleiste der Instanz `hole` aufgerufen.

Beachten Sie im folgenden Beispiel, dass sich der Programmieraufwand durch die Aktion `with` und durch Verwenden der Methoden des MovieClip-Objektes beim Übermitteln von mehreren Anweisungen erheblich verringert:

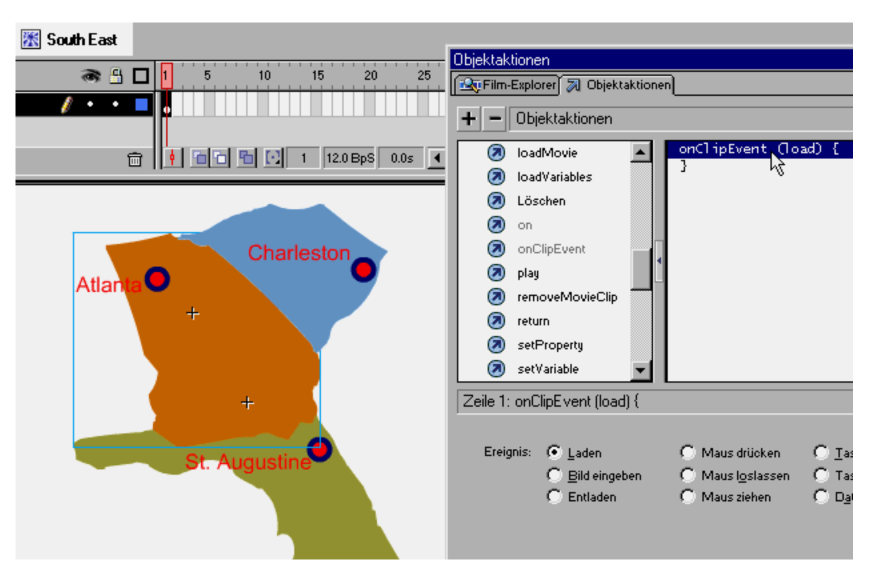
```
with (myMovieClip) {
   _x -= 10;
   _y += 10;
   gotoAndPlay(3);
}
```

Weitere Informationen über die Aktion `tellTarget` finden Sie im *Flash Benutzerhandbuch.*

Zuordnen einer Aktion oder Methode

Aktionen und Methoden können einer Schaltfläche oder einem Bild in einer Zeitleiste oder einer Instanz einer Filmsequenz zugeordnet werden.

Wenn Sie einer Filmsequenzinstanz eine Aktion oder Methode zuordnen möchten, müssen Sie einen `onClipEvent`-Handler definieren. Alle einer Instanz zugewiesenen Aktionen werden in den `onClipEvent`-Handler eingebunden und nach dem Auslösen des Handlers ausgeführt. Die Aktion `onClipEvent` wird durch Zeitleistenereignisse (wie das Laden eines Films) oder Benutzerereignisse (wie Mausklicks oder Tastedrücke) ausgelöst. So löst `onClipEvent(mouseMove)` z.B. immer dann eine Aktion aus, wenn der Benutzer die Maus bewegt.



Die Aktion `onClipEvent` wird einer Instanz auf der Bühne zugeordnet. Die `onClipEvent`-Ereignisse werden im Bedienfeld „Aktionen“ im Parameterfenster aufgeführt.

Laden und Entladen zusätzlicher Filme

Mit der Aktion bzw. Methode `loadMovie` können Sie zusätzliche Filme wiedergeben, ohne den Flash Player zu schließen, sowie zwischen Filmen wechseln, ohne eine weitere HTML-Seite öffnen zu müssen. Außerdem können mit `loadMovie` Variablen an ein CGI-Skript gesendet werden, das eine SWF-Datei als CGI-Ausgabe erzeugt. Wenn Sie einen Film laden, können Sie eine Zielebene oder eine Filmsequenz angeben, in die der Film geladen wird.

Die Aktion bzw. Methode `unloadMovie` entfernt einen zuvor mit `loadMovie` geladenen Film. Das ausdrückliche Entladen eines Filmes mit `unloadMovie` stellt einen weichen Übergang zwischen Filmen sicher und verringert den vom Flash Player benötigten Speicherplatz. Verwenden Sie die Aktion `loadMovie` für die folgenden Aufgaben:

- Abspielen mehrerer als SWF-Dateien vorliegender Werbeeinblendungen hintereinander. Fügen Sie hierzu am Ende jeder SWF-Datei die Aktion `oadMovie` ein, um den nächsten Film zu laden.
- Entwickeln einer verzweigten Oberfläche, über die Benutzer zwischen mehreren SWF-Dateien wählen können.
- Erstellen einer Navigationsoberfläche, bei der über Steuerelemente auf Ebene 0 andere Ebenen geladen werden. Beim Laden von Ebenen erzielen Sie weichere Übergänge als beim Laden von neuen HTML-Seiten in einen Browser.

Ändern von Position und Darstellung einer Filmsequenz

Wenn Sie während der Wiedergabe die Eigenschaften einer Filmsequenz ändern möchten, können Sie die Aktion `setProperty` verwenden oder eine Anweisung schreiben, in der einer Eigenschaft ein Wert zugewisen wird. Beim Laden eines Filmes in ein Ziel übernimmt der geladene Film die Eigenschaften der als Ziel angegebenen Filmsequenz. Sobald der Film geladen wurde, können Sie diese Eigenschaften ändern.

Einige der Eigenschaften sind *schreibgeschützt*. Sie können die Werte dieser Eigenschaften abfragen, nicht jedoch neu setzen. Die Werte aller nicht schreibgeschützten Eigenschaften können durch eine entsprechend geschriebene Anweisung gesetzt werden. In der folgenden Anweisung wird der Wert der Eigenschaft „_alpha“ der Filmsequenzinstanz `wheel` gesetzt. Die Filmsequenz ist ein untergeordnetes Element der Instanz `car`.

```
car.wheel._alpha = 50;
```

Sie können außerdem Anweisungen schreiben, mit denen der Wert einer Filmsequenzeigenschaft abgefragt wird. In der folgenden Anweisung wird z. B. der Wert der Eigenschaft `_xmouse` auf der Hauptzeitleiste ermittelt und die Eigenschaft `_x` der Instanz `customCursor` auf diesen Wert gesetzt:

```
onClipEvent(enterFrame){
   customCursor._x = _root._xmouse;
}
```

Sie können auch die Funktion `getProperty` verwenden, um die Eigenschaften einer Filmsequenz abzufragen.

Die Eigenschaften _x, _y, _rotation, _xscale, _yscale, _height, _width, _alpha und _visible werden durch Veränderungen im übergeordneten Element der Filmsequenz beeinflusst und beeinflussen wiederum sämtliche untergeordneten Elemente der Sequenz. Bei den Eigenschaften _focusrect, _highquality, _quality und _soundbuftime handelt es sich um globale Eigenschaften, die nur der Zeitleiste auf Ebene 0 zugeordnet sind. Alle anderen Eigenschaften sind jeder Filmsequenz oder geladenen Ebene zugeordnet. In der folgenden Tabelle finden Sie eine Auflistung sämtlicher Filmsequenzeigenschaften:

Eigenschaften			
_alpha	_highquality	_totalframes	_xscale
_currentframe	_name	_url	_y
_droptarget	_quality	_visible	_ymouse
_focusrect	_rotation	_width	_yscale
_framesloaded	_soundbuftime	_x	
_height	_target	_xmouse	

Ziehen von Filmsequenzen

Mit der Aktion bzw. Methode startDrag geben Sie eine Filmsequenz während der Wiedergabe eines Filmes für Ziehoperationen frei. Dies kann z.B. bei verschiebbaren Filmsequenzen in einem Spiel, beim Implementieren von Drag & Drop-Funktionen sowie bei anpassbaren Oberflächen, Bildlaufleisten und Schiebereglern nützlich sein.

Eine Filmsequenz kann so lange gezogen werden, bis die Freigabe durch stopDrag aufgehoben oder mit startDrag eine andere Filmsequenz ausgewählt wird. Es kann jeweils nur eine Filmsequenz gezogen werden.

Für komplexere Ziehoperationen bei Filmsequenzen können Sie die Eigenschaft _droptarget verwenden. So können Sie z.B. die Eigenschaft _droptarget verwenden um festzustellen, ob ein Film zu einer bestimmten Filmsequenz verschoben wurde (z.B. einer Filmsequenz mit einem „Abfalleimer"), und dann eine andere Aktion auslösen. Siehe „Verwenden von if-Anweisungen“ auf Seite 75 und „Verwenden von Operatoren zum Ändern von Werten in Ausdrücken“ auf Seite 65.

Duplizieren und Entfernen von Filmen

Sie können Filmsequenzinstanzen während der Wiedergabe Ihres Filmes mit `duplicateMovieClip` duplizieren und mit `removeMovieClip` entfernen. Mit der Aktion bzw. Methode `duplicateMovieClip` wird dynamisch eine neue Instanz der Filmsequenz erstellt, dieser einen neuen Instanznamen zugewisen und eine Tiefe für die Instanz festgelegt. Eine duplizierte Filmsequenz beginnt immer bei Bild 1, auch wenn sich die ursprüngliche Filmsequenz beim Duplizieren an einer anderen Bildposition befand, und liegt über allen vordefinierten Filmsequenzen der Zeitleiste. Variablen werden nicht in die duplizierte Filmsequenz kopiert.

Zum Löschen einer mit `duplicateMovieClip` erstellten Filmsequenz verwenden Sie `removeMovieClip`. Duplizierten Filmsequenzen werden auch gelöscht, wenn die übergeordnete Filmsequenzen gelöscht werden.

Anhängen von Filmsequenzen

Sie können eine Kopie einer Filmsequenz aus einer Bibliothek abrufen und mit Hilfe der Methode `attachMovie` als Teil Ihres Filmes abspielen. Mit dieser Methode wird eine zusätzliche Filmsequenz in Ihre Filmsequenz geladen und die geladene Sequenz während der Wiedergabe des Filmes abgespielt.

Um eine Filmsequenz mit der Methode `attachMovie` anhängen zu können, müssen Sie der anzuhängenden Sequenz im Dialogfeld „Eigenschaften Symbolverknüpfung“ einen eindeutigen Name zuweisen.

So bennenen Sie eine Filmsequenz, damit sie gemeinsam genutzt werden kann:

1. Wählen Sie in der Filmbibliothek die Filmsequenz aus, die angehängt werden soll.
2. Wählen Sie im Fenster „Bibliothek“ im Menü „Optionen“ die Option „Verknüpfung“ aus.
3. Wählen Sie unter „Verknüpfung“ die Option „Dieses Symbol exportieren“ aus.
4. Geben Sie im Dialogfeld „Eigenschaften Symbolverknüpfung“ unter „Bezeichner“ einen Namen für die Filmsequenz ein. Dieser Name muss anders lauten als der Symbolname in der Bibliothek.
5. Klicken Sie auf „OK“.

So hängen Sie eine Filmsequenz an eine andere Filmsequenz an:

1. Geben Sie im Bedienfeld „Aktionen“ das Ziel an, an das Sie die Filmsequenz anhängen möchten.
2. Wählen Sie in der Werkzeugliste erst das MovieClip-Objekt und anschließend die Methode `attachMovie` aus.

3 Geben Sie die folgenden Argumente an:

- Geben Sie für `idName` den Bezeichnernamen an, den Sie im Dialogfeld „Eigenschaften Symbolverknüpfung“ festgelegt haben.
- Geben Sie für `newName` einen Instanznamen für die angehängte Filmsequenz ein, damit Sie diese als Ziel für Operationen festlegen können.
- Geben Sie für `depth` die Ebene ein, auf der der duplizierte Film an die Filmsequenz angehängt werden soll. Jeder angehängte Film verfügt über eine eigene Ebenenanordnung, wobei Ebene 0 der Ebene des ursprünglichen Filmes entspricht. Angehängte Filmsequenzen liegen immer über der ursprünglichen Filmsequenz.

  Beispiel:

```
myMovieClip.attachMovie("calif", "california", 10 );
```

Erstellen von Smart-Filmsequenzen

Bei einer Smart-Filmsequenz handelt es sich um eine Filmsequenz mit bereits definierten Sequenzparametern, die jedoch geändert werden können. Die Parameter werden an Aktionen in der Smart-Filmsequenz übergeben, die das Verhalten der Sequenz beeinflussen.

Sie erstellen eine Smart-Filmsequenz, indem Sie einem Filmsequenzsymbol in der Bibliothek Sequenzparameter zuweisen. In die Smart-Filmsequenz schreiben Sie ActionScript-Anweisungen, die diese Parameter verarbeiten (ähnlich den Argumenten in einer Funktionsdefinition). Sie können eine Smart-Filmsequenz auf der Bühne auswählen und im Bedienfeld "Sequenzparameter" die Werte der Parameter ändern. Während der Wiedergabe werden die im Bedienfeld festgelegten Werte an die Smart-Filmsequenz übermittelt, bevor die Aktionen im Film ausgeführt werden.

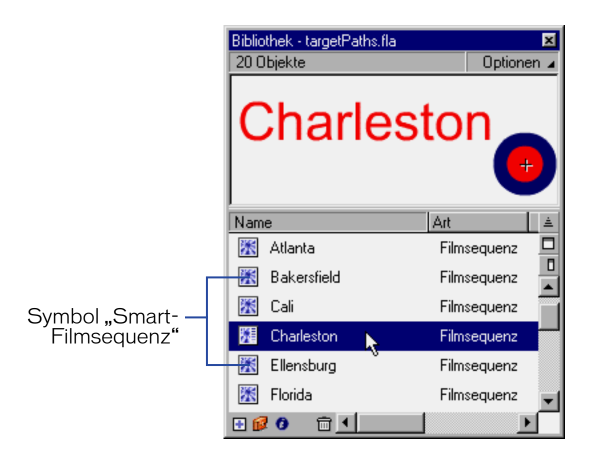
Smart-Filmsequenzen ermöglichen die Weitergabe komplexer Flash Elemente vom Programmierer an den Designer. Der Programmierer schreibt variablenabhängige Aktionen in der Smart-Filmsequenz, über die die Sequenz und der Film gesteuert werden. Der Designer kann dann im Bedienfeld „Sequenzparameter“ die Werte dieser Variablen ändern, ohne das Bedienfeld „Aktionen“ öffnen zu müssen.

Sie können Smart-Filmsequenzen zum Erstellen von Oberflächenelementen verwenden, z. B. für Optionsschaltflächen, Popup-Menüs, Werkzeugtipps, Umfragen, Spiele und Avatare. Smart-Filmsequenzen bieten sich für alle Filmsequenzen an, die Sie ohne Änderungen im Skript auf unterschiedliche Arten wiederverwenden möchten.

Bei Bedarf können Sie in Flash zusätzlich eine angepasste Oberfläche für das Bedienfeld „Sequenzparameter“ erstellen und so die Bedienbarkeit für Designer verbessern, die später die Sequenz bearbeiten werden.

Definieren von Sequenzparametern

Bei Sequenzparametern handelt es sich um Daten, die beim Laden einer Filmsequenz in einem Film an die Sequenz übergeben werden. Die Sequenzparameter werden beim Erstellen eines Filmes definiert. Die Parameter können dann in Aktionen verwendet werden, um die Darstellung und das Verhalten der Smart-Filmsequenz während der Wiedergabe zu ändern. Filmsequenzen mit definierten Sequenzparametern werden im Fenster „Bibliothek“ durch ein spezielles Symbol gekennzeichnet.

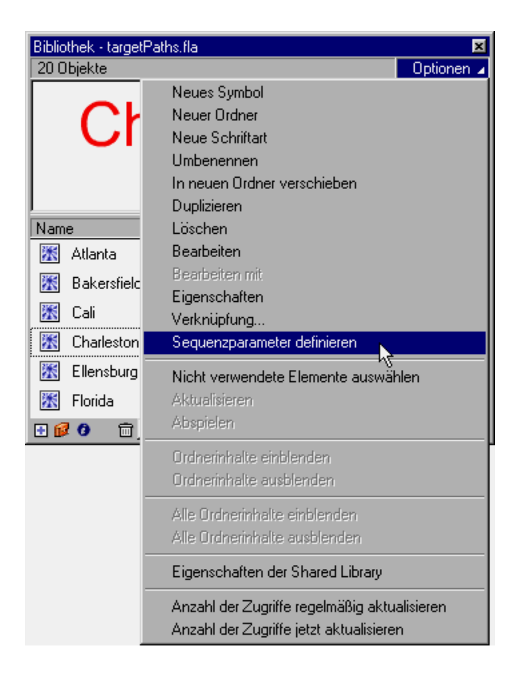


So definieren Sie Sequenzparameter für eine Filmsequenz:

1 Wählen Sie ein Filmsequenzsymbol aus Ihrer Filmbibliothek aus, und führen Sie einen der folgenden Schritte durch, um das Dialogfeld „Sequenzparameter“ zu öffnen:

- Klicken Sie mit der rechten Maustaste (Windows) bzw. drücken Sie CTRL (Macintosh), und wählen Sie im Kontextmenü die Option „Sequenzparameter definieren“ aus.

- Wählen Sie oben rechts im Fenster „Bibliothek“ im Menü „Optionen“ die Option „Sequenzparameter definieren“.



2 Verwenden Sie die Steuerelemente im Dialogfeld „Sequenzparameter“ wie folgt:

- Klicken Sie auf die Schaltfläche „Hinzufügen“ (+), wenn Sie ein neues Name/Wert-Paar hinzufügen oder einem ausgewählten Name/Wert-Paar zusätzliche Parameter hinzufügen möchten.
- Klicken Sie auf die Schaltfläche „Minus“ (-), um ein Name/Wert-Paar zu löschen.
- Verwenden Sie die Pfeiltasten, um die Reihenfolge der Parameter in der Liste zu ändern.
- Mit einem Doppelklick können Sie ein Feld auswählen, um einen Wert einzugeben.

3 Geben Sie unter „Name“ einen eindeutigen Bezeichner für den Parameter ein.

4 Wählen Sie unter „Typ“ aus dem Popup-Menü den Datentyp für den Parameter aus:

- Wählen Sie „Standard“, wenn der Parameter eine Zeichenfolge oder eine Zahl enthalten soll.

- Wählen Sie „Array“, wenn Sie eine dynamische Elementliste mit variabler Größe wünschen.
- Wählen Sie „Objekt“ aus, um mehrere verwandte Elemente mit Namen und Werten zu deklarieren, z. B. ein Punktobjekt mit *x*- und *y*-Elementen.
- Wählen Sie „Liste“ aus, wenn Sie die Auswahl auf bestimmte Eingaben eingrenzen möchten, z. B. `wahr` und `falsch` oder `Rot`, `Grün` und `Blau`.

5 Wählen Sie unter „Wert“ aus dem Popup-Menü den Standardwert aus, auf den der Parameter gesetzt werden soll.

6 Wenn Sie eine benutzerdefinierte Oberfläche für das Bedienfeld „Sequenzparameter“ verwenden möchten, führen Sie einen der folgenden Schritte aus:

- Geben Sie im Feld „Verknüpfung mit benutzerdefinierter Oberfläche“ einen relativen Pfad zu der SWF-Datei ein, die die benutzerdefinierte Oberfläche enthält.
- Klicken Sie auf den Ordner „Verknüpfung mit benutzerdefinierter Oberfläche“, und suchen Sie nach der SWF-Datei für die benutzerdefinierte Oberfläche.

Siehe „Erstellen einer angepassten Oberfläche“ auf Seite 138.

7 Geben Sie unter „Beschreibung“ Anmerkungen zu den einzelnen Parametern ein. Die Anmerkungen werden als Info im Bedienfeld „Sequenzparameter“ angezeigt.

Sie können unter „Beschreibung“ beliebige Informationen für spätere Benutzer der Smart-Filmsequenz einfügen. Fügen Sie beispielsweise eine Erläuterung zu den von Ihnen definierten Methoden ein.

8 Wählen Sie „Sperren in Instanz“ aus, wenn Sie die Namen der Parameter im Bedienfeld „Sequenzparameter“ gegen eine Umbenennung durch Benutzer sperren möchten.

Das Aktivieren dieser Sperre wird empfohlen.

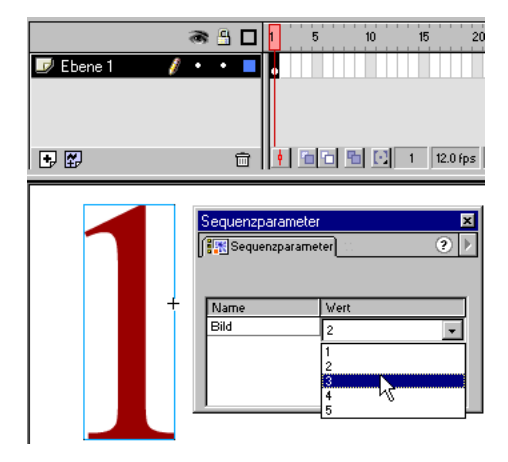
9 Klicken Sie auf „OK“.

Setzen von Sequenzparametern

Sie können Aktionen in die Smart-Filmsequenz einbinden, die die definierten Parameter verwenden und anhand dieser das Verhalten der Smart-Filmsequenz beeinflussen. In einem einfachen Beispiel wird ein Sequenzparameter mit dem Namen `Frame` definiert und wie im folgenden Beispiel von einem Skript innerhalb der Smart-Filmsequenz verwendet:

```
onClipEvent(load){
   gotoAndStop(Frame);
}
```

Anschließend können Sie die Smart-Filmsequenz auf der Bühne auswählen und im Bedienfeld „Sequenzparameter“ den Wert für den Parameter `Frame` setzen, um ein anderes Bild abspielen zu lassen.



So setzen Sie die Parameter für eine Smart-Filmsequenz:

1 Wählen Sie eine Smart-Filmsequenz auf der Bühne aus.

   Bei Smart-Filmsequenzen handelt es sich um Filmsequenzen; daher wird im Erstellungsmodus nur das erste Bild angezeigt.

2 Wählen Sie „Fenster“ > „Bedienfelder“ > „Sequenzparameter“, um das Bedienfeld „Sequenzparameter“ zu öffnen.

3 Führen Sie im Bedienfeld „ Sequenzparameter“ einen der folgenden Schritte durch:

- Doppelklicken Sie auf das Feld „Wert“, um es auszuwählen, und geben Sie einen Wert für jeden Parameter ein.

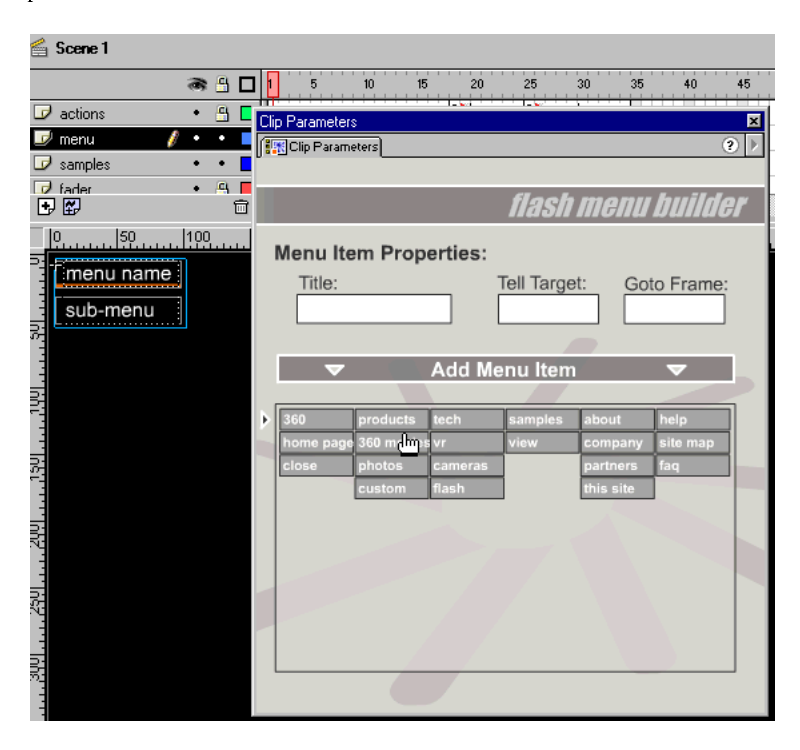
  Wenn der Parameter als Liste deklariert wurde, wird ein Popup-Menü angezeigt.

- Wenn eine benutzerdefinierte Oberfläche erstellt wurde, verwenden Sie die hier vorhandenen Oberflächenelemente.

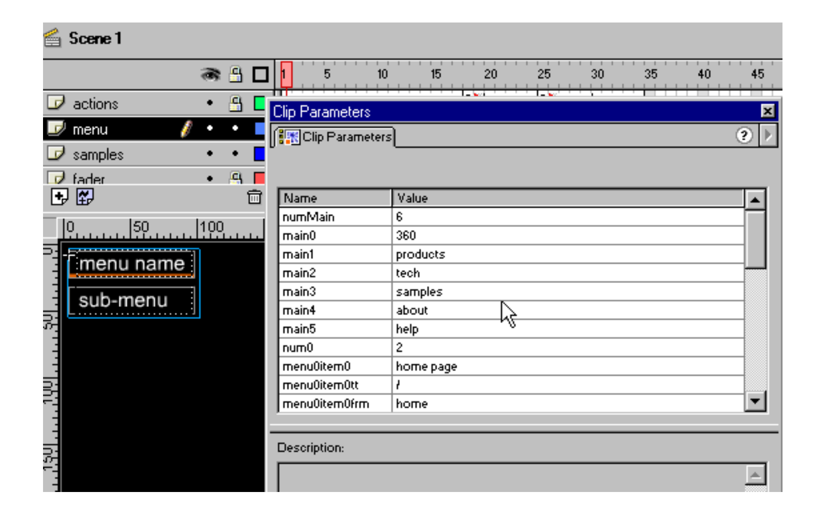
4 Wählen Sie „Steuerung“ > „Film testen“, um die Änderungen im Verhalten der Smart-Filmsequenz zu überprüfen.

Erstellen einer angepassten Oberfläche

Bei einer angepassten Oberfläche handelt es sich um einen Flash Film, in dem Sie Werte eingeben können, die an eine Smart-Filmsequenz übergeben werden. Die angepasste Oberfläche ersetzt hierbei die Oberfläche des Bedienfeldes „Sequenz-parameter".



Das Bedienfeld „Sequenzparameter" mit einer angepassten Oberfläche.



Dieselbe Smart-Filmsequenz ohne eine angepasste Oberfläche im Bedienfeld „Sequenzparameter“.

Alle Werte, die Sie über eine angepasste Oberfläche eingeben, werden vom Bedienfeld „Sequenzparameter“ über eine Zwischensequenz (die Austauschfilsequenz) an die Smart-Filmsequenz in der angepassten Oberfläche weitergegeben. Die Austauschfilmsequenz muss den Instanznamen `xch` aufweisen. Wenn eine angepasste Oberfläche im Dialogfeld „Sequenzparameter definieren“ ausgewählt wurde, übergibt die Instanz der Smart-Filmsequenz die definierten Parameter an die Filmsequenz `xch`. Alle in der angepassten Oberfläche neu eingegebenen Werte werden nach `xch` kopiert und an die Smart-Filmsequenz zurückgegeben.

Die Sequenz `xch` muss auf der Hauptzeitleiste des Oberflächenfilmsplatziert werden und dauerhaft geladen sein. Die Filmsequenz `xch` sollte ausschließlich die an die Smart-Filmsequenz zu übergebenden Werte enthalten. Fügen Sie keine Grafiken, andere Filmsequenzen oder ActionScript-Anweisungen ein. `xch` dient ausschließlich als Behälter, mit dem die Werte weitergegeben werden. Sie können Objekte der obersten Ebene, z. B. Arrays und Objekte, mit der Sequenz `xch` übergeben. Sie sollten jedoch keine verschachtelten Arrays oder Objekte übergeben.

So erstellen Sie eine angepasste Oberfläche für eine Smart-Filmsequenz:

1 Wählen Sie „Datei“ > „Neu“, um einen neuen Flash Film zu erstellen.

2 Wählen Sie „Einfügen” > “Neues Symbol”, um die Austauschfilmsequenz zu erstellen.

3 Erstellen Sie eine neue Ebene mit der Bezeichnung „Austauschsequenz“.

4 Während die Ebene „Austauschsequenz“ ausgewählt ist, ziehen Sie die Austauschfilmsequenz aus dem Fenster „Bibliothek“ auf die Bühne in Bild 1.

5 Wählen Sie die Austauschfilmsequenz auf der Bühne aus, wählen Sie „Fenster“ > „Bedienfelder“ > „Instanz“, und geben Sie den Namen `xch` ein.

6 Erstellen Sie die Oberflächenelemente, über die der Autor später die Sequenzparameter festlegt. Dies können z. B. Popup-Menüs, Optionsschaltflächen oder Drag & Drop-Menüelemente sein.

7 Verwenden Sie die Aktion `set variable`, um die Variablen- und Objektwerte in die Instanz `xch` zu kopieren.

   Wenn z. B. eine Schaltfläche als Oberflächenelement dient, kann diese mit einer Aktion verbunden werden, die den Wert der Variablen `vertical` setzt und an `xch` übergibt, wie im folgenden Beispiel gezeigt:

```
on (release){
    _root.xch.vertical = true;
}
```

8 Exportieren Sie den Film als SWF-Datei.

Damit Sie die angepasste Oberfläche (SWF-Datei) in einer Smart-Filmsequenz verwenden können, müssen Sie beide im Dialogfeld „Sequenzparameter definieren“ der Bibliothek verknüpfen, die die Smart-Filmsequenz enthält. Es empfiehlt sich, die SWF-Datei im selben Verzeichnis wie die FLA-Datei abzulegen, die die Smart-Filmsequenz enthält. Wenn Sie die Smart-Filmsequenz in einer anderen Datei wiederverwenden oder diese an einen anderen Entwickler weitergeben, dürfen die relativen Speicherorte der Smart-Filmsequenz und der SWF-Datei für die angepasste Oberfläche nicht geändert werden.



KAPITEL 5
Integrieren von Flash in Webanwendungen

Flash Filme können Informationen in Remote-Dateien schreiben und aus diesen lesen. Verwenden Sie die Aktion `loadVariables` und `getURL`, um Variablen zu senden und zu laden. Um einen Flash Player-Film von einem Remote-Speicherort zu laden, verwenden Sie die Aktion `loadMovie`. Um XML-Daten zu senden oder zu laden, verwenden Sie das XML- oder das XMLSocket-Objekt. Mit den vordefinierten Methoden für XML-Objekte können Sie XML-Daten strukturieren.

Sie können Flash Formulare mit Standardoberflächenelementen wie z.B Textfeldern und Popup-Menüs erstellen, in denen die Daten erfasst werden, die an eine Server-Anwendung übertragen werden sollen.

Wenn Flash so erweitert werden soll, dass Meldungen aus der Host-Umgebung des Filmes, z.B aus dem Flash Player oder einer JavaScript-Funktion in einem Webbrowser, empfangen oder an diese gesendet werden sollen, verwenden Sie die Anweisung `fscommand` und bestimmte Flash Player-Methoden.

Senden und Laden von Variablen an bzw. aus einer Remote-Datei

Bei einem Flash Film handelt es sich um ein Fenster, in dem ähnlich wie auf HTML-Seiten Informationen aufgezeichnet und angezeigt werden. Im Gegensatz zu HTML-Seiten können Flash Filme jedoch dauerhaft in den Browser geladen werden und regelmäßig mit neuen Informationen aktualisiert werden, ohne dass der Film erneut geladen werden muss. Verwenden Sie Flash Aktionen und Objektmethoden, um Informationen an Server-Skripts, Textdateien und XML-Dateien zu senden oder aus diesen zu empfangen.

Server-Skripts können Informationen aus einer Datenbank abrufen und zwischen der Datenbank und einem Flash Film übertragen. Die Server-Skripte können in einer Vielzahl unterschiedlicher Sprachen verfasst sein: zu den am häufigsten eingesetzten gehören Perl, ASP (Microsoft Active Server Pages) und PHP.

Durch Speichern von Informationen in einer Datenbank und Abrufen dieser Daten können Sie dynamische und personalisierte Inhalte in Ihren Film einbinden. Sie können z. B ein Mitteilungsbrett, persönliche Profile für Benutzer oder einen Einkaufskorb erstellen, mit dem die Einkäufe eines Benutzers festgehalten und so die Vorlieben dieses Benutzers ermittelt werden.

Sie können verschiedene ActionScript-Aktionen und Objektmethoden verwenden, um Informationen in einen Film zu übertragen oder aus diesem abzurufen. Bei jeder Aktion und Methode wird ein Protokoll verwendet, um die Informationen zu übertragen. Außerdem ist ein bestimmtes Format für die Informationsübertragung erforderlich.

Bei den folgenden Aktionen wird das HTTP- oder HTTPS-Protokoll verwendet, um Informationen im URL-kodierten Format zu übertragen: `getURL`, `loadVariables` und `loadMovie`.

Bei den folgenden Methoden wird das HTTP- oder HTTPS-Protokoll verwendet, um Informationen als XML zu übertragen: `XML.send`, `XML.load` und `XML.sendAndLoad`.

Bei den folgenden Methoden wird eine TCP/IP-Socket-Verbindung erstellt und verwendet, um Informationen als XML zu übertragen: `XMLSocket.connect` und `XMLSocket.send`.

Erläuterungen zur Sicherheit

Beim Abspielen eines Flash Filmes in einem Webbrowser können in den Film nur Daten aus Dateien geladen werden, die sich auf Servern in derselben Subdomäne befinden. Auf diese Weise wird verhindert, dass über einen Flash Film Informationen von fremden Servern heruntergeladen werden können.

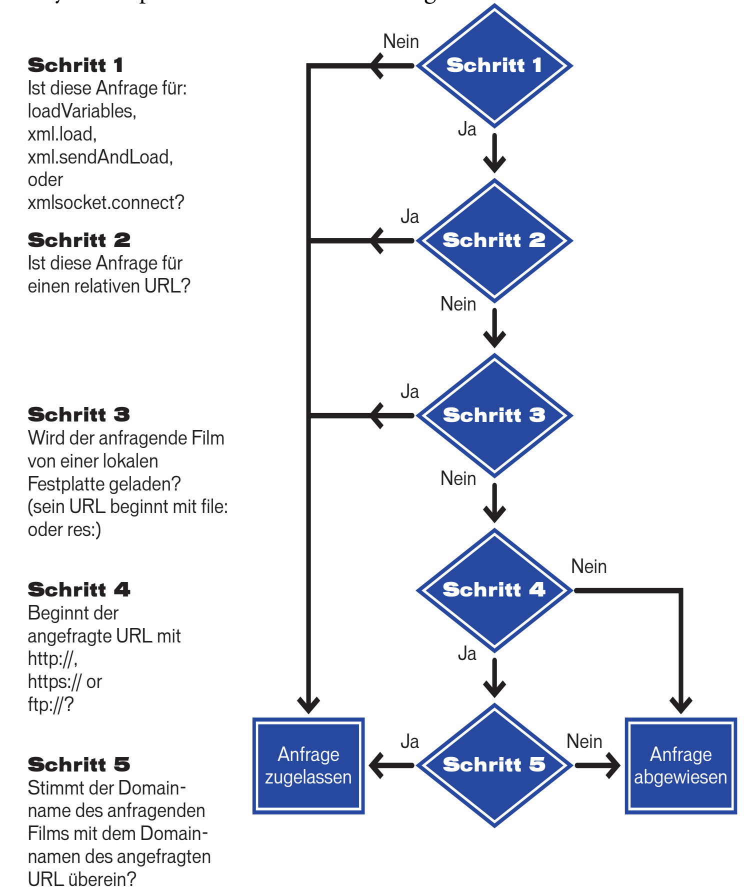
Um die Subdomäne eines URL zu ermitteln, der aus einer oder zwei Elementen besteht, verwenden Sie die gesamte Domäne:

Domäne	Subdomäne
http://macromedia	macromedia
http://macromedia.com	macromedia.com

Um die Subdomäne eines URL zu ermitteln, der aus mehr als zwei Elementen besteht, entfernen Sie die letzte Ebene:

Domäne	Subdomäne
http://x.y.macromedia.com	y.macromedia.com
http://www.macromedia.com	macromedia.com

Die folgende Abbildung veranschaulicht, nach welchem Verfahren der Flash Player überprüft, ob eine HTTP-Anfrage erlaubt wird oder nicht:



Wenn Sie das XMLSocket-Objekt zum Erstellen einer Socket-Verbindung mit einem Server verwenden, muss als Portnummer 1024 oder höher angegeben werden. (Ports mit kleineren Nummern werden im Allgemeinen für Telnet, FTP, das World Wide Web oder Finger genutzt.)

Flash verwendet standardisierte Browser-, HTTP- und HTTPS-Sicherheitsfunktionen. Grundsätzlich unterstützt Flash dieselben Sicherheitsfunktionen, die normalerweise auch unter HTML zur Verfügung stehen. Sie sollten also auch die gleichen Regeln anwenden, die für das Einrichten sicherer HTML-Websites gelten. Wenn Sie in Flash z. B sichere Kennwörter verwenden möchten, müssen Sie Ihre Kennwortauthentisierung mit einer Anfrage bei einem Webserver einrichten.

Wenn Sie ein Kennwort erstellen möchten, verwenden Sie ein Textfeld, in dem der Benutzer das Kennwort eingibt. Übermitteln Sie das Kennwort mit der Aktion `loadVariables` oder der Methode `XML.sendAndLoad` an einen Server. Verwenden Sie hierbei einen HTTPS-URL mit der Methode `POST`. Auf dem Webserver kann dann ermittelt werden, ob dieses Kennwort auch gültig ist. Auf diese Weise taucht das Kennwort niemals in der SWF-Datei auf.

Überprüfen, ob Daten geladen wurden

Mit Ausnahme von `XMLSocket.send` arbeitet jede Aktion und Methode, die Daten in einen Film lädt, *asynchron*. Die Ergebnisse der Aktion werden also zu einem unbekannten Zeitpunkt zurückgegeben.

Bevor Sie geladene Daten in einem Film verwenden können, müssen Sie prüfen, ob die Daten tatsächlich geladen wurden. Sie können z. B nicht in einem Skript Variablen laden und die Werte dieser Variablen im gleichen Skript bearbeiten. Im folgenden Skript können Sie erst auf die Variable `lastFrameVisited` zugreifen, nachdem sichergestellt ist, dass die Variable aus der Datei `myData.txt` geladen wurde:

```
loadVariables("myData.txt", 0);
gotoAndPlay(lastFrameVisited);
```

Für jede Aktion und Methode steht ein bestimmtes Verfahren zur Verfügung, mit dem geprüft werden kann, welche Daten geladen wurden. Bei den Aktionen `loadVariables` und `loadMovie` können Sie Informationen in eine Filmsequenz laden und das Ereignis `data` der Aktion `onClipEvent` verwenden, um ein Skript auszuführen. Wenn Sie die Aktion `loadVariables` zum Laden der Daten verwenden, wird die Aktion `onClipEvent(data)` ausgeführt, sobald die letzte Variable geladen wurde. Wenn Sie die Daten mit Hilfe der Aktion `loadMovie` laden, wird die Aktion `onClipEvent(data)` jedes Mal beim Laden eines Filmsegments in den Flash Player ausgeführt.

Die folgende Schaltflächenaktion z. B lädt die Variablen aus der Datei `myData.txt` in die Filmsequenz `loadTargetMC`:

```
on(release){
   loadVariables("myData.txt", _root.loadTargetMC);
}
```

Eine der Instanz `loadTargetMC` zugewiesene Aktion verwendet die Variable `lastFrameVisited`, die aus der Datei `myData.txt` geladen wird. Die folgende Aktion wird erst ausgeführt, nachdem alle Variablen, einschließlich `lastFrameVisited`, geladen wurden:

```
onClipEvent(data) {
   goToAndPlay(lastFrameVisited);
}
```

Bei den Methoden `XML.load` und `XMLSocket.connect` können Sie einen Handler definieren, der die Daten beim Empfang verarbeitet. Bei einem Handler handelt es sich um eine Eigenschaft des XML- oder XMLSocket-Objektes, der Sie eine von Ihnen erstellte Funktion zuweisen. Die Handler werden automatisch beim Empfang der entsprechenden Informationen aufgerufen. Verwenden Sie `XML.onLoad` für das XML-Objekt und `XMLSocket.onConnect` für das XMLSocket-Objekt.

Weitere Informationen finden Sie unter „Verwenden des XML-Objekts“ auf Seite 147 und „Verwenden des XMLSocket-Objekts“ auf Seite 152.

Verwenden von loadVariables, getURL und loadMovie

Die Aktionen `loadVariables`, `getURL` und `loadMovie` kommunizieren mit den jeweiligen Server-Skripts unter Verwendung des HTTP-Protokolls. Jede Aktion sendet alle Variablen der Zeitleiste, mit der die Aktion verbunden ist. Die Aktionen reagieren hierbei folgendermaßen auf die Antworten:

- `getURL` gibt alle Informationen an ein Browser-Fenster zurück, nicht an den Flash Player.
- `loadVariables` lädt Variablen in eine festgelegte Zeitleiste im Flash Player.
- `loadMovie` lädt einen Film in eine festgelegte Ebene im Flash Player.

Bei den Aktionen `loadVariables`, `getURL` und `loadMovie` können Sie mehrere Argumente angeben:

- *`URL`* bezeichnet die Datei, in der sich die Remote-Variablen befinden.
- *`Ort`* bezeichnet die Ebene oder das Ziel innerhalb des Filmes, an die oder das die Variablen gesendet werden.

  Weitere Informationen über Ebenen und Ziele finden Sie unter „Erläuterungen zur Verwendung mehrerer Zeitleisten“ auf Seite 112.

  Anmerkung: Die Aktion `getURL` unterstützt dieses Argument nicht.

- *`Variablen`* legt die HTTP-Methode fest (`GET` oder `POST`), nach der die Variablen gesendet werden.

Wenn Sie z.B die Punktewertungen für ein Spiel verfolgen möchten, können Sie diese auf einem Server speichern und anschließend über die Aktion `loadVariables` immer dann in den Film laden, wenn eine Person das entsprechende Spiel gespielt hat. Die Aktion sieht in etwa so aus:

```
loadVariables("http://www.mySite.com/scripts/high_score.php",
_root.scoreClip, GET)
```

Hierdurch werden Variablen aus dem PHP-Skript mit dem Namen `high_score.php` in die Filmsequenzinstanz `scoreClip` geladen. Das Laden erfolgt über die HTTP-Methode `GET`.

Alle mit der Aktion `loadVariables` geladenen Variablen müssen im Standard-MIME-Format „application/x-www-urlformencoded" vorliegen (einem von CGI-Skripts verwendeten Standardformat). In der im URL-Argument der Aktion `loadVariables` angegebenen Datei müssen die Variablen und zugehörigen Werte im genannten Format vorliegen, damit diese in Flash gelesen werden können.

Die Datei kann eine beliebige Anzahl an Variablen enthalten. Variablen- und Wertepaare müssen durch ein kaufmännisches Und-Zeichen (`&`), Wörter innerhalb eines Wertes durch ein Plus (+) voneinander getrennt sein. Mit dem folgenden Ausdruck werden z.B mehrere Variablen definiert:

```
highScore1=54000&playerName1=rockin+good&highScore2=53455&playerN
ame2=bonehelmet&highScore3=42885&playerName3=soda+pop
```

Weitere Informationen zu `loadVariables`, `getURL` und `loadMovie` finden Sie unter den entsprechenden Einträgen in Kapitel 7, „Referenz zu ActionScript".

Erläuterungen zu XML

XML (*Extensible Markup Language*) entwickelt sich zum Standard für den Austausch von strukturierten Daten in Internetanwendungen. Sie können in Flash eine Datenintegration mit Servern realisieren, bei denen XML-Technologie eingesetzt wird, und so anspruchsvolle Anwendungen entwickeln, wie beispielsweise ein Chat- oder ein Broker-System.

Wie in HTML können Sie in XML Tags verwenden, um einen Textabschnitt *auszuzeichnen*. In HTML verwenden Sie vordefinierte Tags, um die gewünschte Textdarstellung in einem Webbrowser festzulegen (mit dem Tag `<b>` wird ein Textabschnitt z.B fett formatiert). In XML definieren Sie Tags, mit denen der Typ eines Datensegments angegeben wird (z.B `<password>VerySecret</password>`). Mit XML werden Struktur und Darstellungsform der Informationen voneinander getrennt. Dadurch kann ein XML-Dokument in unterschiedlichen Umgebungen verwendet werden.

Jedes XML-Tag wird als *Knoten* (oder Element) bezeichnet. Jeder Knoten verfügt über einen Typ (1: XML-Element oder 3: Textknoten). Zusätzlich können Elemente Attribute aufweisen. Ein in einem anderen Knoten verschachtelter Knoten wird *untergeordneter Knoten* genannt. Diese hierarchische Baumstruktur der Knoten wird als XML DOM (Document Object Model) bezeichnet. Es kann mit dem JavaScript DOM verglichen werden, mit dem die Struktur der Elemente in einem Webbrowser festgelegt wird.

Im folgenden Beispiel ist `<PORTFOLIO>` der übergeordnete Knoten. Er besitzt keine Attribute und beinhaltet den untergeordneten Knoten `<HOLDING>`, für den die Attribute `SYMBOL`, `QTY`, `PRICE` und `VALUE` festgelegt wurden:

```
<PORTFOLIO>
  <HOLDING SYMBOL="RICH"
           QTY="75"
           PRICE="245,50"
           VALUE="18412,50" />
</PORTFOLIO>
```

Verwenden des XML-Objekts

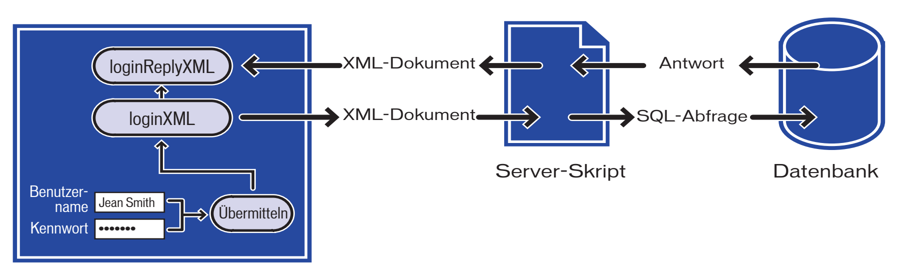
Sie können die Methoden des XML-Objektes in ActionScript (z. B `appendChild`, `removeNode` und `insertBefore`) verwenden, um XML-Daten in Flash für die Übertragung an einen Server zu strukturieren oder heruntergeladene XML-Daten zu bearbeiten und zu interpretieren.

Verwenden Sie die folgenden XML-Objektmethoden, um XML-Daten mit Hilfe der HTTP-Methode `POST` an einen Server zu senden oder von diesem zu laden:

- `load` lädt XML-Daten von einem URL herunter und platziert sie in einem ActionScript-XML-Objekt.
- `send` übermittelt ein XML-Objekt an einen URL. Alle zurückgegebenen Informationen werden an ein anderes Browser-Fenster gesendet.
- `sendAndLoad` sendet ein XML-Objekt an einen URL. Alle zurückgegebenen Informationen werden in einem ActionScript-XML-Objekt abgelegt.

Sie könnten z. B ein Broker-System für den Handel mit Wertpapieren erstellen, bei dem alle Informationen (Benutzernamen, Kennwörter, Sitzungs-IDs, Portfolio und Transaktionsinformationen) in einer Datenbank gespeichert werden.

Das Server-Skript, über das die Daten zwischen Flash und der Datenbank übertragen werden, liest und schreibt die Daten im XML-Format. Mit ActionScript können Sie die im Flash Film erfassten Informationen (z.B Benutzername und Kennwort) in ein XML-Objekt umwandeln und die Daten als XML-Dokument an das Server-Skript senden. Außerdem können Sie das vom Server zurückgegebene XML-Dokument mit ActionScript in ein XML-Objekt laden und dieses in einem Film verwenden.



Datenfluss und Umwandlung der Daten zwischen einem Flash Player-Film, einem Server-Skriptdokument und einer Datenbank.

Für die Kennwortprüfung des Broker-Systems sind zwei Skripts erforderlich: eine im ersten Bild definierte Funktion und ein Skript, das die XML-Objekte erstellt und sendet, die mit der Schaltfläche „Übermitteln“ im Formular verbunden sind.

Wenn ein Benutzer im Flash Film die entsprechenden Informationen in die Textfelder für die Variablen `username` und `password` eingibt, müssen diese Variablen vor dem Weiterleiten an den Server in XML umgewandelt werden. Im ersten Abschnitt des Skripts werden die Variablen in ein neu erstelltes XML-Objekt mit der Bezeichnung `loginXML` geladen. Sobald ein Benutzer auf die Schaltfläche „Übermitteln“ klickt, wird das Objekt `loginXML` in eine XML-Zeichenfolge umgewandelt und an den Server gesendet.

Das folgende Skript ist der Schaltfläche „Übermitteln“ zugeordnet. Lesen Sie die mit den Zeichen // eingeleiteten Kommentarzeilen, um ein besseres Verständnis der Skripts zu erlangen:

```
on (release) {
  // A. XML-Dokument mit einem Element "LOGIN" erstellen
  loginXML = new XML();
  loginElement = loginXML.createElement("LOGIN");
  loginElement.attributes.username = username;
  loginElement.attributes.password = password;
  loginXML.appendChild(loginElement);

  // B. Ein XML-Objekt erstellen, das die Antwort des Servers 
aufnimmt
  loginReplyXML = new XML();
  loginReplyXML.onLoad = onLoginReply;

  // C. Element "LOGIN" an den Server senden
  //    und die Antwort in "loginReplyXML" ablegen
  loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",
                       loginReplyXML);
}
```

Im ersten Abschnitt des Skripts wird das folgende XML-Konstrukt erzeugt, sobald ein Benutzer auf die Schaltfläche „Übermitteln“ klickt:

```
<LOGIN USERNAME="JeanSmith" PASSWORD="VerySecret" />
```

Der Server empfängt das XML-Konstrukt, erzeugt eine XML-Antwort und sendet diese zurück an den Flash Film. Wenn das Kennwort akzeptiert wurde, antwortet der Server wie folgt:

```
<LOGINREPLY STATUS="OK" SESSION="rnr6f7vkj2oe14m7jkkycilb" />
```

Dieses XML-Konstrukt beinhaltet das Attribut `SESSION`, das eine zufällig generierte und eindeutige Sitzungs-ID enthält. Diese wird während der gesamten Sitzung für die Kommunikation zwischen dem Client und dem Server verwendet. Wenn das Kennwort abgelehnt wird, gibt der Server folgende Meldung zurück:

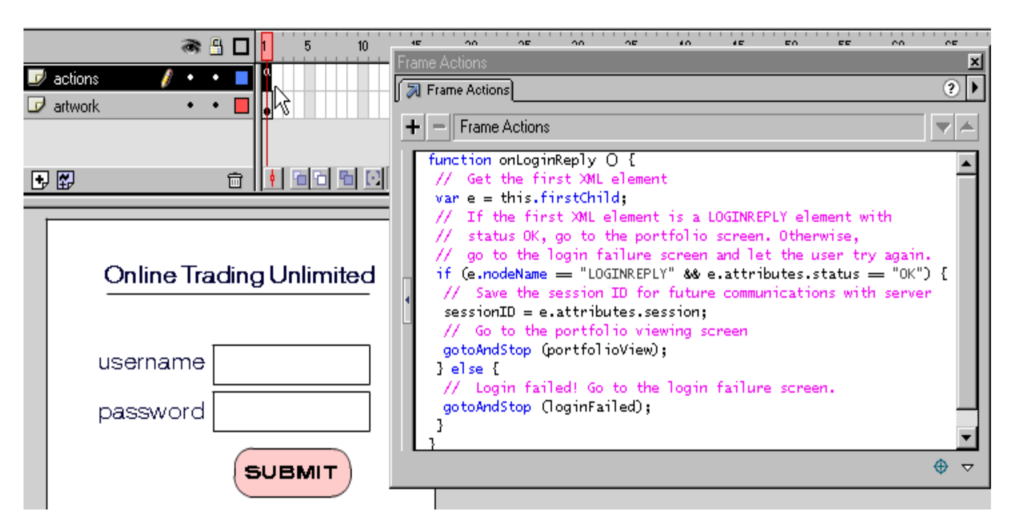
```
<LOGINREPLY STATUS="FAILED" />
```

Der XML-Knoten `LOGINREPLY` muss in ein leeres XML-Objekt im Flash Film geladen werden. Die folgende Anweisung erstellt das XML-Objekt `loginreplyXML`, in dem der XML-Knoten abgelegt wird:

```
// B. Ein XML-Objekt erstellen, das die Antwort des Servers 
aufnimmt
loginReplyXML = new XML();
loginReplyXML.onLoad = onLoginReply;
```

In der zweiten Anweisung wird dem Handler `loginReplyXML.onLoad` die Funktion `onLoginReply` zugewiesen.

Das XML-Element `LOGINREPLY` wird asynchron empfangen (ähnlich den Daten der Aktion `loadVariables`) und in das Objekt `loginReplyXML` geladen. Beim Empfang der Daten wird die Methode `onLoad` des `loginReplyXML`-Objektes aufgerufen. Sie müssen die Funktion `onLoginReply` definieren und dem Handler `loginReplyXML.onLoad` zuweisen, damit dieser das Element `LOGINREPLY` verarbeiten kann. Die Funktion `onLoginReply` wird dem Bild zugewiesen, das die Schaltfläche „Übermitteln" beinhaltet.



Definieren der Funktion `onLoginReply` im ersten Bild des Filmes.

Die Funktion `onLoginReply` wird im ersten Bild des Filmes definiert. Lesen Sie die mit den Zeichen `//` eingeleiteten Kommentarzeilen, um ein besseres Verständnis der Skripts zu erlangen:

```
function onLoginReply() {
  // Erstes XML-Element zuweisen
  var e = this.firstChild;
  // Wenn das erste XML-Element ein LOGINREPLY-Element mit
  // Status "OK" ist, zum Portfolio-Bildschirm wechseln. Andern-
falls
  // zum Bildschirm für fehlgeschlagene Anmeldung wechseln, damit
der Benutzer es erneut versuchen kann.
  if (e.nodeName == "LOGINREPLY" && e.attributes.status == "OK") {
// Sitzungs-ID für nachfolgende Kommunikation mit dem Server spe-
ichern
    sessionID = e.attributes.session;
// Zum Portfolio-Bildschirm wechseln
    gotoAndStop("portfolioView");
  } else {
    // Anmeldung fehlgeschlagen!  Zum Bildschirm für fehlge-
schlagene Anmeldung wechseln
    gotoAndStop("loginFailed");
  }
}
```

In der ersten Zeile dieser Funktion, `var e = this.firstChild`, wird das Schlüsselwort `this` verwendet, um auf das XML-Objekt `loginReplyXML` zu verweisen, das XML enthält und gerade vom Server geladen wurde. Sie können `this` verwenden, da `onLoginReply` als `loginReplyXML.onLoad` aufgerufen wurde. Obwohl es sich bei `onLoginReply` um eine einfache Funktion zu handeln scheint, wird diese wie eine Methode von `loginReplyXML` behandelt und verarbeitet.

Um den Benutzernamen und das Kennwort als XML an den Server zu senden und eine XML-Antwort zurück in den Flash Film zu laden, verwenden Sie die Methode `sendAndLoad` wie folgt:

```
// C. Element "LOGIN" an den Server senden
//    und die Antwort in "loginReplyXML" ablegen
  loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",
loginReplyXML);
```

Weitere Informationen zu XML-Methoden finden Sie unter den entsprechenden Einträgen in Kapitel 7, „Referenz zu ActionScript“.

Anmerkung: Bei diesem Entwurf handelt es sich lediglich um ein Beispiel, und es kann nicht garantiert werden, dass hierbei tatsächlich eine angemessene Sicherheitsebene erreicht wird. Wenn Sie ein sicheres System mit Kennwortschutz implementieren möchten, sollten Sie genauestens mit Fragen der Netzwerksicherheit vertraut sein.

Verwenden des XMLSocket-Objekts

Unter ActionScript ist ein vordefiniertes XMLSocket-Objekt verfügbar, über das Sie eine ständige Verbindung mit einem Server öffnen können. Durch die Socket-Verbindung kann der Server Informationen an den Client weitergeben, sobald diese Informationen zur Verfügung stehen. Ohne eine ständige Verbindung müsste der Server auf eine HTTP-Anfrage warten. Durch eine solche offene Verbindung werden Verzögerungen vermieden. Es wird deshalb üblicherweise für Echtzeitanwendungen wie Chats genutzt. Die Daten werden als Zeichenfolge über die Socket-Verbindung gesendet und müssen im XML-Format vorliegen. Mit dem XML-Objekt können Sie eine Struktur für die Daten festlegen.

Um eine Socket-Verbindung herzustellen, müssen Sie eine Server-Anwendung erstellen, die auf die Anfrage für eine Socket-Verbindung wartet und eine Antwort an den Flash Film sendet. Sie können eine solche Server-Anwendung in einer Programmiersprache wie z. B Java verfassen.

Für die Übertragung von XML-Daten über eine Socket-Verbindung zu oder von einem Server werden die Methoden `connect` und `send` des ActionScript-XML-Socket-Objektes verwendet. Mit der Methode `connect` richten Sie eine Socket-Verbindung mit einem Port des Webservers ein. Mit der Methode `send` übergeben Sie ein XML-Objekt an den in der Socket-Verbindung festgelegten Server.

Sobald Sie die Methode `connect` des XMLSocket-Objektes aufrufen, öffnet der Flash Player eine TCP/IP-Verbindung mit dem Server und hält diese Verbindung offen, bis eine der folgenden Bedingungen eintritt:

- Die Methode `close` des XMLSocket-Objektes wird aufgerufen.
- Es sind keine Verweise mehr auf das XMLSocket-Objekt vorhanden.
- Der Flash Player wird beendet.
- Die Verbindung wird unterbrochen (z. B wenn die Modemverbindung getrennt wird).

Im folgenden Beispiel wird eine XML-Socket-Verbindung hergestellt, über die Daten aus dem XML-Objekt `myXML` gesendet werden. Lesen Sie die mit den Zeichen `//` eingeleiteten Kommentarzeilen, um ein besseres Verständnis der Skripts zu erlangen:

```
// neues XMLSocket-Objekt erzeugen
sock = new XMLSocket();
// Methode "connect" aufrufen, um eine Verbindung mit Port 1024
// des Servers mit dem URL herzustellen
sock.connect("http://www.myserver.com", 1024);
// Funktion definieren, die dem "sock"-Objekt zugewiesen wird, mit
dem die Antwort
// des Servers verarbeitet wird. Bei erfolgreichem Verbindungsauf-
bau das "myXML"-Objekt senden. Wenn nicht, eine Fehlermeldung in
einem Textfeld ausgeben.
function onSockConnect(success){
   if (success){
      sock.send(myXML);
   } else {
      msg="Fehler beim Verbinden mit "+serverName;
   }
}
// Funktion "onSockConnect" der Eigenschaft "onConnect" zuweisen
sock.onConnect = onSockConnect;
```

Weitere Informationen finden Sie unter dem Eintrag „XMLSocket" in Kapitel 7, „Referenz zu ActionScript".

Erstellen von Formularen

Flash Formulare stellen eine Erweiterung der Interaktivität dar. Sie sind eine Kombination aus Schaltflächen, Filmen und Textfeldern, mit denen Informationen an andere Anwendungen auf einem lokalen oder einem Remote-Server weitergegeben werden können. Alle allgemeinen Formularelemente (zum Beispiel Optionsschaltflächen, Dropdown-Listen und Kontrollkästchen) können als Filme oder Schaltflächen erstellt und der allgemeinen Gestaltung Ihrer Website angepasst werden. Hierbei zählen Eingabetextfelder zu den am häufigsten verwendeten Formularelementen.

Als Beispiele für Formulare, in denen solche Elemente üblicherweise verwendet werden, sind z. B Chat-Oberflächen, Bestellformulare und Suchmasken zu nennen. So kann ein Flash Formular Adressinformationen aufnehmen und an andere Anwendungen senden, die diese Informationen in E-Mail-Nachrichten oder Datenbankdateien zusammenstellen. Auch ein einzelnes Textfeld wird als Formular behandelt und kann zum Erfassen von Benutzereingaben oder Anzeigen von Ergebnissen verwendet werden.

Für Formulare sind zwei Hauptkomponenten erforderlich: die Flash Oberflächenelemente, aus denen die Formulare bestehen, und eine Server-Anwendung oder ein Client-Skript, mit der oder dem die vom Benutzer eingegebenen Informationen verarbeitet werden. Die folgenden Schritte zeigen die grundlegende Vorgehensweise zum Erstellen eines Formulars in Flash.

So erstellen Sie ein Formular:

1 Positionieren Sie entsprechend dem gewünschten Layout Oberflächenelemente im Film.

Sie können Oberflächenelemente aus der allgemeinen Bibliothek „Buttons-Advanced“ verwenden oder eigene Elemente erstellen.

2 Legen Sie im Bedienfeld „Textoptionen“ Textfelder als Eingabefelder fest, und weisen Sie jedem Textfeld einen eindeutigen Variablennamen zu.

Weitere Informationen zum Erstellen von editierbaren Textfeldern finden Sie im *Flash Benutzerhandbuch*.

3 Weisen Sie eine Aktion zu, mit der die Daten gesendet, geladen oder sowohl gesendet als auch geladen werden.

Erstellen eines Suchformulars

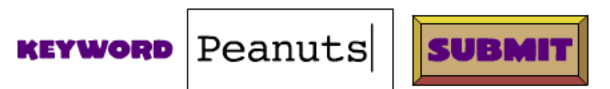
Ein Suchfeld mit einer Schaltfläche „Übermitteln“ ist ein Beispiel für ein einfaches Suchformular. Als Einführung in das Erstellen von Formularen finden Sie im folgenden Beispiel Anweisungen zum Erstellen einer Suchmaske unter Verwendung der Aktion `getURL`. Durch die Eingabe der erforderlichen Informationen können Benutzer ein Schlüsselwort an eine Suchmaschine übergeben, welche auf einem Remote-Webserver an einem anderen Standort ausgeführt wird.

So erstellen Sie ein einfaches Suchformular:

1 Erstellen Sie eine Schaltfläche für die Übermittlung der eingegebenen Daten.

2 Erstellen Sie auf der Bühne eine Beschriftung, ein leeres Textfeld und eine Instanz der Schaltfläche.

Auf dem Bildschirm sollte das folgende Bild ausgegeben werden:



3 Wählen Sie das Textfeld aus, und wählen Sie „Fenster“ > „Bedienfelder“ > „Textoptionen“.

4 Legen Sie im Bedienfeld „Textoptionen“ die folgenden Optionen fest:

- Wählen Sie im Popup-Menü die Option „Eingabetext“ aus.
- Wählen Sie „Rand“.
- Geben Sie einen Variablennamen an.

Anmerkung: Für einige Suchmaschinen sind besondere Variablennamen erforderlich. Weitere Informationen finden Sie in der Website der Suchmaschine.

5 Wählen Sie die Schaltfläche auf der Bühne aus, und wählen Sie „Fenster“ > „Aktionen“.

   Das Bedienfeld „Objektaktionen“ wird angezeigt.

   Anmerkung: Im Menü „Fenster“ zeigt ein Häkchen neben der Option „Aktionen“ an, dass das Bedienfeld geöffnet ist.

6 Ziehen Sie die Aktion `getURL` aus der Werkzeugleiste in das Skriptfenster.

7 Stellen Sie im Parameterfenster die folgenden Optionen ein:

- Geben Sie im Feld „URL“ den URL der Suchmaschine ein.
- Wählen Sie für das Feld „Fenster“ die Option `_blank` aus. Hierdurch wird ein neues Fenster geöffnet, in dem die Suchergebnisse angezeigt werden.
- Geben Sie unter „Variablen“ die Option „Mit GET versenden“ ein.

8 Wählen Sie „Datei“ > „Vorschau für Veröffentlichungen“ > „HTML“, um das Formular zu testen.

Verwenden von Variablen in Formularen

Sie können Variablen in Formularen verwenden, um Benutzereingaben zu speichern. Verwenden Sie editierbare Textfelder oder Aktionen, die Sie bestimmten Schaltflächen in Oberflächenelementen zuweisen, um die Variablenwerte zu setzen. Bei den Elementen in einem Popup-Menü handelt es sich beispielsweise um Schaltflächen mit zugeordneten Aktionen. Mit den Aktionen werden Variablen gesetzt, die das ausgewählte Element wiedergeben. Eingabetextfeldern können Variablennamen zugewiesen werden. Das Textfeld wirkt als Fenster, das den Wert dieser Variablen anzeigt.

Wenn Sie Informationen an ein Server-Skript übergeben oder von diesem empfangen, müssen die im Flash Film verwendeten Variablen mit denen im Skript übereinstimmen. Wenn das Skript z. B eine Variable mit dem Namen `password` erwartet, muss das Textfeld, in dem der Benutzer das Kennwort eingibt, den Variablennamen `password` tragen.

Für einige Skripts sind versteckte Variablen erforderlich. Dies sind Variablen, die dem Benutzer nicht angezeigt werden. Um eine versteckte Variable in Flash zu erstellen, können Sie eine Variable auf einem Bild der Filmsequenz festlegen, das auch die anderen Formularelemente enthält. Die versteckten Variablen werden zusammen mit allen anderen Variablen an das Server-Skript übertragen, die für die Zeitleiste definiert sind, welche die Aktion zum Übermitteln des Formulars enthält.

Überprüfen der eingegebenen Daten

Wenn Sie ein Formular entwickeln, mit dem Variablen an eine Anwendung auf einem Webserver übergeben werden, werden Sie unter Umständen überprüfen wollen, ob die Benutzer die Daten auch korrekt eingeben. Damit können Sie z. B ausschließen, dass Benutzer Text in ein Feld für Telefonnummern eingeben. Verwenden Sie in diesem Fall mehrere `set variable`-Aktionen in Verbindung mit `for` und `if`, um die eingegebenen Daten zu überprüfen.

Im folgenden Beispiel wird überprüft, ob es sich bei den eingegebenen Daten um Zahlen handelt und die Zahlen im Format "-"-"# eingegeben wurden. Wenn die Daten richtig sind, soll die Meldung „Dies ist eine richtige Telefonnummer." angezeigt werden. Wenn die Daten ungültig sind, wird „Diese Telefonnummer ist ungültig." angezeigt.

Um dieses Skript in einem Film zu verwenden, erstellen Sie auf der Bühne zwei Textfelder und wählen für jedes der Felder im Bedienfeld „Textoptionen" die Option „Eingabetext" aus. Ordnen Sie einem der Textfelder die Variable `phoneNumber` und dem anderen die Variable `message` zu. Verbinden Sie die folgende Aktion mit einer Schaltfläche, die Sie auf der Bühne neben den Textfeldern positionieren:

```
on (release) {
   valid = validPhoneNumber(phoneNumber);
   if (valid) {
      message = "Dies ist eine richtige Telefonnummer.";
   } else {
      message = "Diese Telefonnummer ist ungültig.";
   }
   function isdigit(ch) {
      return ch.length == 1 && ch >= '0' && ch <= '9';
   }
   function validPhoneNumber(phoneNumber) {
      if (phoneNumber.length != 12) {
         return false;
      }
      for (var index = 0; index < 12; index++) {
         var ch = phoneNumber.charAt(index);
         if (index == 3 || index == 7) {
            if (ch != "-") {
            return false;
            }
         } else if (!isdigit(ch)) {
            return false;
         }
      }
      return true;
   }
}
```

Erstellen Sie eine Schaltfläche, die in etwa folgende Aktion ausführt, um die Daten zu senden. (Ersetzen Sie die Argumente von getURL durch die entsprechenden Argumente Ihres Filmes.)

```
on (release) {
   if (valid) {
      getURL("http://www.webserver.com", "_self", "GET");
   }
}
```

Weitere Informationen über diese ActionScript-Anweisungen finden Sie unter set, for und if in Kapitel 7, „Referenz zu ActionScript“ auf Seite 173“.

Senden und Empfangen von Meldungen im Flash Player

Wenn Sie Meldungen aus einem Flash Film an dessen Host-Umgebung senden möchten (z. B an einen Webbrowser, einen Director Film oder den Flash Player), verwenden Sie die Aktion fscommand. Dies eröffnet Ihnen größere Möglichkeiten in ihrem Film, da Sie Funktionen des Hosts nutzen können. Sie können z. B eine fscommand-Aktion an eine JavaScript-Funktion in einer HTML-Seite übergeben, mit der ein neues Browser-Fenster mit bestimmten Eigenschaften geöffnet wird.

Wenn Sie einen Flash Player-Film über die Skriptsprache eines Webbrowsers steuern möchten, z. B über JavaScript, VBScript oder Microsoft JScript, können Sie Flash Player-Methoden verwenden. Dies sind Funktionen, die Meldungen aus einer Host-Umgebung an den Flash Film senden. Sie können z. B einen Hyperlink in eine HTML-Seite einbinden, mit dem der Flash Film auf eine bestimmte Bildposition gesetzt wird.

Verwenden von fscommand

Mit der Aktion fscommand wird eine Meldung an ein beliebiges Programm übermittelt, das den Flash Player ausführt. Die Aktion fscommand weist zwei Parameter auf: *Befehl* und *Argumente*. Wenn Sie eine Meldung an eine eigenständige Version des Flash Player senden möchten, müssen Sie vordefinierte Befehle und Argumente verwenden. Mit der folgenden Aktion wird der eigenständige Player z. B angewiesen, den Film in der vollen Bildschirmgröße darzustellen, sobald die Schaltfläche losgelassen wird:

```
on(release){
   fscommand("fullscreen", "true");
}
```

In der folgenden Tabelle werden die zulässigen Werte für die Parameter *Befehl* und *Argumente* der Aktion `fscommand` aufgeführt, mit der Sie den in einem eigenständigen Player (dazu gehören auch Projektoren) wiedergegebenen Film steuern können:

Befehl	*Argumente*	**Funktion**
`quit`	Keine	Schließt den Projektor.
`fullscreen`	`true` oder `false`	Mit `true` wird der Flash Player in den Vollbildmodus gesetzt. Mit `false` wird der Player in die normale Menüansicht zurückgesetzt.
`allowscale`	`true` oder `false`	Mit `false` wird der Player angewiesen, den Film nur in dessen ursprünglicher Größe darzustellen. Mit `true` wird der Film exakt in der Größe des Players dargestellt.
`showmenu`	`true` oder `false`	Mit `true` werden sämtliche Kontextmenüelemente aktiviert. Mit `false` werden alle Elemente des Kontextmenüs außer der Option „Info zu Flash 5...“ ausgeblendet.
`exec`	Pfad der Anwendung	Startet eine Anwendung aus dem Projektor heraus.

Wenn Sie mit `fscommand` eine Meldung an eine Skriptsprache (z. B JavaScript) in einem Webbrowser senden, können Sie zwei beliebige Argumente mit den Parametern *Befehl* und *Argumente* übergeben. Bei diesen Argumenten kann es sich um Zeichenfolgen oder Ausdrücke handeln, die in der JavaScript-Funktion verarbeitet werden, in der die Aktion `fscommand` abgefangen wird.

Eine Aktion `fscommand` ruft die JavaScript-Funktion *moviename*`_DoFSCommand` in der HTML-Seite auf, in welche der Flash Film eingebettet ist. Hierbei ist *moviename* der Name für den Flash Player, der mit dem Attribut `NAME` des Tags `EMBED` oder dem Attribut `ID` des Tags `OBJECT` zugewiesen wurde. Wenn dem Flash Player der Name `myMovie` zugewiesen wurde, muss die JavaScript-Funktion `myMovie_DoFSCommand` aufgerufen werden.

So verwenden Sie die Aktion „fscommand“, um mit JavaScript ein Meldungsfeld aus einem Flash Film in der HTML-Seite zu öffnen:

1 Fügen Sie der HTML-Seite, die den Flash Film enthält, den folgenden JavaScript-Code hinzu:

```
function theMovie_DoFSCommand(command, args) {
    if (command == "messagebox") {
        alert(args);
    }
}
```

Wenn Sie Ihren Film unter Verwendung der Vorlage „Flash with FSCommand“ (einzustellen im Dialogfeld „Einstellungen für Veröffentlichungen“, Register „HTML") veröffentlichen, wird dieser Code automatisch eingefügt. Die Attribute `NAME` und `ID` des Filmes dienen als Dateiname. Für den Dateinamen `myMovie.fla` werden die Attributwerte zum Beispiel auf `myMovie` gesetzt. Weitere Informationen zum Veröffentlichen finden Sie im *Flash Benutzerhandbuch*.

2 Weisen Sie im Flash Film einer Schaltfläche die Aktion `fscommand` zu:

```
fscommand("messagebox", "Dieses Meldungsfeld wurde von Flash
aus aufgerufen.")
```

Sie können auch Ausdrücke für die Aktion `fscommand` und die zugehörigen Argumente verwenden, wie im folgenden Beispiel gezeigt:

```
fscommand("messagebox", "Hallo, " & name & ". Willkommen auf
unserer Website!")
```

3 Wählen Sie „Datei“ > „Vorschau für Veröffentlichungen“ > „HTML“, um den Film zu testen.

Über die Aktion `fscommand` können Meldungen an Macromedia Director gesendet werden, die von Lingo als Zeichenfolgen, Ereignisse oder ausführbarer Lingo-Code interpretiert werden. Wenn es sich um eine Zeichenfolge oder ein Ereignis handelt, müssen Sie entsprechenden Lingo-Code schreiben, der die Meldungen der Aktion `fscommand` empfängt und in Director eine Aktion aufruft. Weitere Informationen finden Sie beim Director Support Center unter http://www.macromedia.com/support/director.

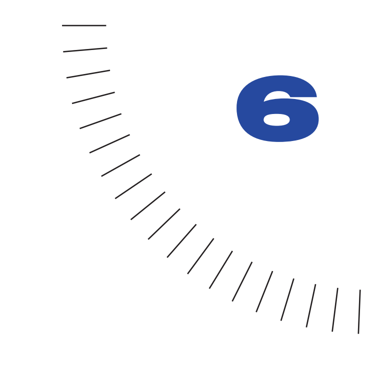
In Visual Basic, Visual C++ und in anderen Programmen, in denen ActiveX-Steuerelemente ausgeführt werden können, sendet `fscommand` ein VB-Ereignis mit zwei Zeichenfolgen, die in der Programmiersprache der Umgebung verarbeitet werden können. Wenn Sie weitere Informationen wünschen, suchen Sie im Flash Support Center unter http://www.macromedia.com/support/flash nach den Schlüsselwörtern `Flash method`.

Erläuterungen zu Flash Player-Methoden

Mit Hilfe der Flash Player-Methoden können Sie einen Film im Flash Player aus Skriptsprachen eines Webbrowsers (z.B JavaScript oder VBScript) heraus steuern. Wie bei anderen Methoden können Sie mit diesen Flash Player-Methoden aus einer anderen Skriptumgebung als ActionScript Aufrufe an Flash Player-Filme senden. Jede Methode besitzt einen Namen, und bei den meisten Methoden werden Argumente benötigt. Mit einem Argument wird ein bestimmter Wert angegeben, der von der Methode verarbeitet wird. Bei den von einigen Methoden durchgeführten Berechnungen werden Werte zurückgegeben, die in der Skriptumgebung verarbeitet werden können.

Zwei unterschiedliche Technologien ermöglichen die Kommunikation zwischen dem Browser und dem Flash Player: LiveConnect (Netscape Navigator 3.0 oder höher unter Windows 95/98/2000/NT oder Power Macintosh) und ActiveX (Microsoft Internet Explorer 3.0 und höher unter Windows 95/98/2000/NT). Obwohl sich die Scripting-Verfahren aller Browser und Sprachen ähneln, stehen bei der Verwendung von ActiveX-Steuerelementen zusätzliche Eigenschaften und Ereignisse zur Verfügung.

Weitere Informationen, einschließlich einer vollständigen Liste der Flash Player-Skriptmethoden, finden Sie, wenn Sie im Flash Support Center unter http://www.macromedia.com/support/flash nach den Schlüsselwörtern `Flash method` suchen.



KAPITEL 6
ActionScript - Fehlerbehandlung

Vor allem bei gemeinsamer Verwendung mehrerer anspruchsvoller Aktionen kann die Komplexität von Flash Filmen stark zunehmen. Wie in jeder anderen Programmiersprache ist es möglich, fehlerhaftes ActionScript zu schreiben, was zu Fehlern in Ihren Skripts führt. Eine gute Programmiertechnik erleichtert die Fehlerbehandlung bei einem unerwarteten Verhalten des Filmes.

Flash stellt Ihnen mehrere Werkzeuge zur Verfügung, mit denen Sie Ihre Filme im Filmtestmodus oder in einem Webbrowser testen können. Im Debugger wird eine hierarchisch aufgebaute Liste der Filmsequenzen angezeigt, die zu diesem Zeitpunkt im Flash Player geladen sind. Sie können sich auch Variablenwerte anzeigen lassen und während der Filmwiedergabe bearbeiten. Im Filmtestmodus werden im Ausgabefenster Fehlermeldungen und Variablen- und Objektlisten angezeigt. Sie können auch die Aktion `trace` in Ihren Skripts verwenden, um Hinweise zur Programmierung und Werte von Ausdrücken an das Ausgabefenster zu senden.

Richtlinien für Erstellung und Fehlerbehandlung

Indem Sie bei der Skripterstellung sorgfältig vorgehen, reduzieren Sie die Zahl der Bugs (Programmierungsfehler) in den Filmen. Die folgenden Richtlinien helfen Ihnen, Probleme zu vermeiden oder ggf. schnell zu beseitigen.

Empfohlene Vorgehensweisen bei der Skripterstellung

Es empfiehlt sich, im Verlauf der Arbeit mehrere Versionen des Filmes zu speichern. Wählen Sie „Datei“ > „Speichern unter“, um in halbstündigen Abständen eine Version unter einem anderen Namen zu speichern. Mit Hilfe dieser Versionshistorie können Sie herausfinden, wann ein Problem zum ersten Mal aufgetreten ist, indem Sie die letzte Datei ermitteln, die das Problem noch nicht aufweist. Auf diese Weise verfügen Sie immer über eine funktionsfähige Version, selbst wenn eine Datei beschädigt wird.

Außerdem sollten Sie frühzeitige und häufige Tests auf allen Zielplattformen durchführen, damit Probleme so schnell wie möglich erkannt werden können. Wählen Sie bei bedeutenden Änderungen und vor jedem Speichern einer Version „Steuerung“ > „Film testen“, um den Film im Filmtestmodus abzuspielen. Im Filmtestmodus wird der Film in einer Version des eigenständigen Players abgespielt.

Wenn Ihr Zielpublikum den Film im Web betrachten wird, müssen Sie den Film unbedingt auch in einem Browser testen. In bestimmten Situationen wie bei der Entwicklung einer Intranetsite kennen Sie Browser und Plattform Ihres Zielpublikums. Wenn Sie jedoch für eine Website entwickeln, sollten Sie Ihren Film mit allen denkbaren Browsern und Plattformen testen.

Es empfiehlt sich, bei der Erstellung folgende Vorgehensweisen zu befolgen:

- Verwenden Sie die Aktion `trace`, um Kommentare an das Ausgabefenster zu senden. (Siehe „Verwenden des Befehls „trace““ auf Seite 172.)
- Verwenden Sie die Aktion `comment`, um Anweisungen einzufügen, die nur im Bedienfeld „Aktionen“ angezeigt werden. (Siehe „Kommentare“ auf Seite 55.)
- Benennen Sie Elemente eines Skripts nach konsistenten Namenskonventionen. Es empfiehlt sich beispielsweise, in Namen keine Leerzeichen zu verwenden. Die Namen von Variablen und Funktionen sollten mit einem Kleinbuchstaben und jedes neue Wort mit einem Großbuchstaben beginnen (`myVariableName`, `myFunctionName`). Namen von Konstruktorfunktionen beginnen mit einem Großbuchstaben (`MyConstructorFunction`). Wählen Sie eine sinnvolle Namenskonvention, und behalten Sie diese konsequent bei.
- Verwenden Sie aussagekräftige Variablennamen, die den Inhaltstyp der Variablen erkennen lässt. Eine Variable, die Informationen über die zuletzt gedrückte Schaltfläche enthält, könnten Sie beispielsweise `lastButtonPressed` nennen. Ein Name wie `foo` gibt keine Hinweise auf den Inhalt der Variablen.
- Als Alternative zum Debugger können Sie bearbeitbare Textfelder in Führungsebenen für die Verfolgung von Variablenwerten verwenden.
- Im Film-Explorer können Sie sich im Filmbearbeitungsmodus die Anzeigeliste und sämtliche Aktionen in einem Film anzeigen lassen. Siehe Flash Hilfe.

- Verwenden Sie die Aktion `for...in`, um die Eigenschaften von Filmsequenzen einschließlich untergeordneter Filmsequenzen zu durchlaufen. Mit den Aktionen `for...in` und `trace` können Sie eine Liste der Eigenschaften an das Ausgabefenster senden. Siehe „Wiederholen einer Aktion“ auf Seite 75.

Verwenden einer Prüfliste für die Fehlerbehandlung

Wie in jeder Skripting-Umgebung gibt es bestimmte Fehler, die Verfassern von Skripts häufig unterlaufen. Die folgende Liste soll Ihnen als Ausgangspunkt für die Fehlerbehandlung bei einem Film dienen:

- Vergewissern Sie sich, dass Sie sich im Filmtestmodus befinden.

  Im Erstellungsmodus funktionieren nur einige einfache Schaltflächen- und Bildaktionen, beispielsweise `gotoAndPlay` und `stop`. Wählen Sie zum Aktivieren dieser Aktionen „Steuerung“ > „Bildaktionen aktivieren“ oder „Steuerung“ > „Einzelbildaktionen aktivieren“.

- Vergewissern Sie sich, dass Bildaktionen auf mehreren Ebenen nicht zu Konflikten führen.
- Wenn Sie im normalen Modus mit dem Bedienfeld „Aktionen“ arbeiten, muss die Anweisung auf Ausdruck gesetzt sein.

  Wenn Sie einer Aktion einen Ausdruck übergeben und das Feld „Ausdruck“ nicht aktiviert ist, wird der Wert als Zeichenfolge übergeben. Siehe „Verwenden von Operatoren zum Ändern von Werten in Ausdrücken“ auf Seite 65.

- Stellen Sie sicher, dass nicht mehrere ActionScript-Elemente denselben Namen tragen.

  Sie sollten allen Variablen, Funktionen, Objekten und Eigenschaften eindeutige Namen geben. Lokale Variablen stellen jedoch eine Ausnahme dar: Sie müssen nur innerhalb ihres Gültigkeitsbereichs eindeutig sein und werden häufig als Zähler verwendet. Siehe „Festlegen des Gültigkeitsbereichs einer Variablen“ auf Seite 61.

Weitere Tipps für die Fehlerbehandlung bei einem Flash Film erhalten Sie im Flash Support Center unter http://www.macromedia.com/support/flash.

Verwenden des Debuggers

Mit Hilfe des Debuggers können Sie Fehler in einem Film finden, während er im Flash Player abgespielt wird. Sie können sich die Anzeigeliste der Filmsequenzen und geladenen Filme anzeigen lassen und außerdem die Werte von Variablen und Eigenschaften ändern, um korrekte Werte festzulegen. Anschließend können Sie zum Skript zurückkehren und dieses so bearbeiten, dass die gewünschten Ergebnisse erzielt werden. Um den Debugger verwenden zu können, müssen Sie den Flash Debug Player ausführen, eine besondere Version des Flash Players.

Der Flash Debug Player wird automatisch mit der Flash 5-Erstellungsanwendung installiert. Er ermöglicht es Ihnen, die Anzeigeliste, Variablennamen und zugehörige Werte sowie Eigenschaftennamen und zugehörige Werte in den Debugger der Flash Erstellungsanwendung zu laden.

So lassen Sie sich den Debugger anzeigen:

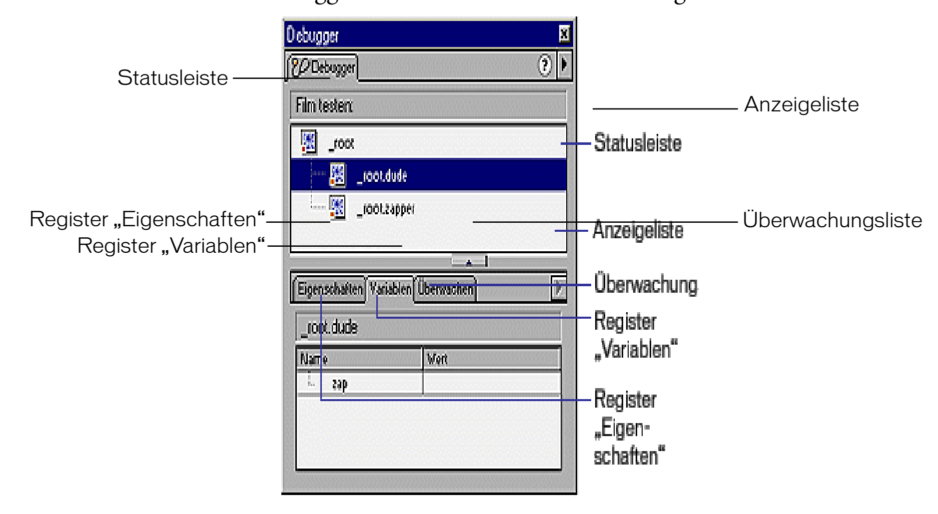
Wählen Sie „Fenster“ > „Debugger“.

Der Debugger wird im inaktiven Status geöffnet. In der Anzeigeliste werden Informationen erst angezeigt, wenn vom Flash Player ein Befehl ausgegeben wird.

So aktivieren Sie den Debugger im Filmtestmodus:

Wählen Sie „Steuerung"> „Fehlersuche“.

Der Debugger wird nun im aktiven Zustand geöffnet.



Aktivieren von Debugging in einem Film

Beim Exportieren eines Flash Player-Filmes können Sie das Debugging in dem Film aktivieren und ein Debugging-Kennwort erstellen. Wenn Sie Debugging nicht aktivieren, wird der Debugger nicht aktiviert.

Wie in JavaScript oder HTML sind im Prinzip alle Variablen in Client-Action-Script für den Benutzer sichtbar. Zur sicheren Speicherung von Variablen müssen Sie diese an eine Server-Anwendung senden und nicht im Film speichern.

Als Flash Entwickler möchten Sie jedoch möglicherweise weitere Informationen geheim halten, beispielsweise die Struktur von Filmsequenzen. Sie können Ihren Film mit einem Debugger-Kennwort veröffentlichen, um zu gewährleisten, dass nur autorisierte Benutzer Ihren Film mit dem Flash Debug Player ansehen können.

So aktivieren Sie Debugging und erstellen ein Kennwort:

1 Wählen Sie „Datei“ > „Einstellungen für Veröffentlichungen“.

2 Klicken Sie auf das Register „Flash“.

3 Aktivieren Sie „Fehlersuche zugelassen“.

4 Zum Festlegen eines Kennwortes geben Sie im Feld „Kennwort“ ein Kennwort ein.

   Ohne dieses Kennwort können keine Informationen in den Debugger geladen werden. Wenn Sie das Feld „Kennwort“ leer lassen, ist kein Kennwort erforderlich.

So aktivieren Sie den Debugger in einem Webbrowser:

1 Klicken Sie mit der rechten Maustaste (Windows) bzw. bei gedrückter Ctrl-Taste (Macintosh), um das Kontextmenü des Flash Debug Players zu öffnen.

2 Wählen Sie „Debugger“.

Anmerkung: Mit dem Debugger können Sie immer nur einen Film überwachen. Um den Debugger verwenden zu können, muss Flash geöffnet sein.



Kontextmenü des Flash Debug Players

Erläuterungen zur Statusleiste

Nach Aktivierung des Debuggers werden in der Statusleiste der URL oder der lokale Dateipfad des Filmes angezeigt. Je nach Wiedergabeumgebung ist der Flash Player unterschiedlich implementiert. In der Statusleiste des Debuggers wird der Typ des Flash Players angezeigt, in dem der Film abgespielt wird:

- „Filmtestmodus“
- „Eigenständiger Player“
- „Netscape-Plug-In“

  Das Netscape-Plug-In wird für Netscape Navigator auf Windows- und Macintosh-Systemen und für Microsoft Internet Explorer auf Macintosh-Systemen verwendet.

- ActiveX-Steuerelement

  Das ActiveX-Steuerelement wird für Internet Explorer unter Windows verwendet.

Erläuterungen zur Anzeigeliste

Bei aktiviertem Debugger wird eine laufend aktualisierte Anzeigeliste der Filmsequenzen angezeigt. Sie können die Strukturen erweitern oder ausblenden, um sich alle auf der Bühne befindlichen Filmsequenzen anzeigen zu lassen. Wenn im Film Filmsequenzen hinzugefügt oder entfernt werden, werden die Änderungen unmittelbar in der Anzeigeliste dargestellt. Sie können die Größe der Anzeigeliste ändern, indem Sie den horizontalen Fensterteiler verschieben oder die untere rechte Ecke ziehen.

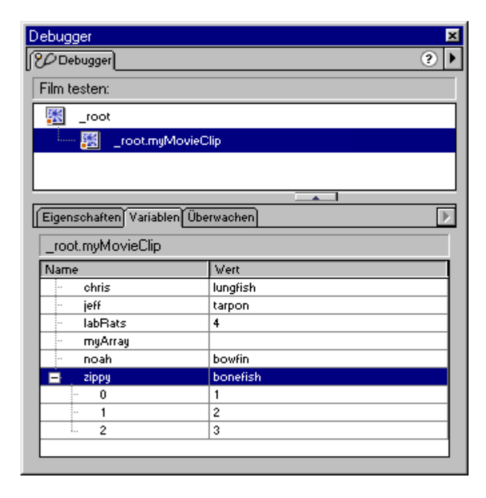
Anzeigen und Ändern von Variablen

Im Debugger werden auf dem Register „Variablen“ die Namen und Werte aller im Film verwendeten Variablen angezeigt. Wenn Sie den Wert einer Variablen auf dem Register „Variablen“ ändern, wirkt sich das während des Abspielens auf den Film aus. Wenn Sie beispielsweise die Kollisionserkennung in einem Spiel testen möchten, könnten Sie den Variablenwert eingeben, mit dem ein Ball am gewünschten Ort in der Nähe einer Wand platziert wird.

So lassen Sie sich eine Variable anzeigen:

1. Wählen Sie in der Anzeigeliste die Filmsequenz aus, die die Variable enthält.
2. Klicken Sie auf das Register „Variablen“.

Die Anzeigeliste wird während des Abspielens des Filmes automatisch aktualisiert. Wenn eine Filmsequenz bei einem bestimmten Bild aus dem Film entfernt wird, wird diese ebenfalls aus der Anzeigeliste im Debugger entfernt. Dabei werden Variablenname und -wert entfernt.



So ändern Sie einen Variablenwert:

Markieren Sie den Wert, und geben Sie einen neuen Wert ein.

Dabei muss es sich um einen konstanten Wert handeln (beispielsweise „`Hallo`“, `3523` oder „`http://www.macromedia.com`“), nicht um einen Ausdruck (beispielsweise `x + 2` oder `eval(„name:" +i)`). Der Wert kann eine Zeichenfolge (jeder in Anführungszeichen (`""`) eingeschlossene Wert), eine Zahl oder ein Boolean (`true` oder `false`) sein.

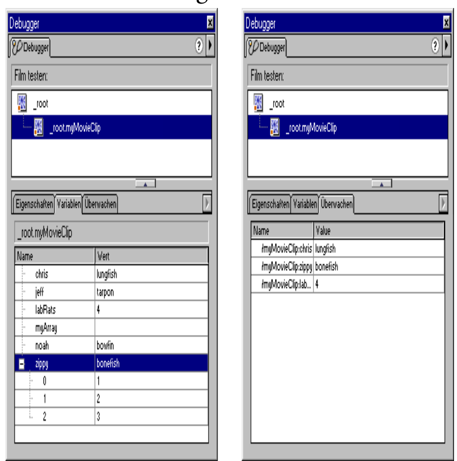
Objekt- und Arrayvariablen werden auf dem Register „Variablen“ angezeigt. Klicken Sie auf die Schaltfläche „Hinzufügen“ (+), um sich die entsprechenden Eigenschaften und Werte anzeigen zu lassen. Sie können in die Wertefelder jedoch keine Objekt- und Arraywerte (beispielsweise `{name: "Ich bin ein Objekt"}` oder `[1, 2, 3]`) eingeben.

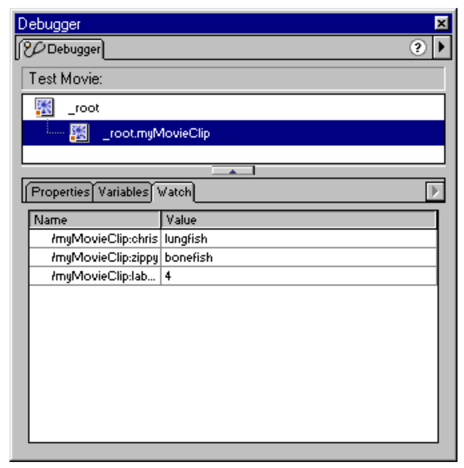
Anmerkung: Im Filmtestmodus können Sie mit der Aktion `trace` den Wert eines Ausdrucks ausgeben. Siehe „Verwenden des Befehls „trace““ auf Seite 172.

Verwenden der Überwachungsliste

Wenn Sie eine Reihe wichtiger Variablen übersichtlich überwachen möchten, können Sie diese zur Anzeige in der Überwachungsliste markieren. In der Überwachungsliste wird der absolute Pfad der Variablen und des Wertes angezeigt. Sie können dort auch einen neuen Variablenwert eingeben.

Der Überwachungsliste können nur Variablen hinzugefügt werden, jedoch keine Eigenschaften oder Funktionen.





Für die Überwachungsliste markierte Variablen und Variablen in der Überwachungsliste.

Wählen Sie eines der folgenden Verfahren, um der Überwachungsliste Variablen hinzuzufügen:

- Klicken Sie auf dem Register „Variablen“ mit der rechten Maustaste (Windows) oder bei gedrückter Ctrl-Taste (Macintosh) auf eine markierte Variable, und wählen Sie im Kontextmenü „Überwachen“. Neben der Variablen wird ein blauer Punkt angezeigt.
- Klicken Sie auf dem Register „Überwachen“ mit der rechten Maustaste (Windows) oder bei gedrückter Ctrl-Taste (Macintosh), und wählen Sie im Kontextmenü „Hinzufügen“. Geben Sie Variablenname und -wert in die Felder ein.

So entfernen Sie Variablen aus der Überwachungsliste:

Klicken Sie auf dem Register „Überwachen“ mit der rechten Maustaste (Windows) oder bei gedrückter Ctrl-Taste (Macintosh), und wählen Sie im Kontextmenü „Entfernen“.

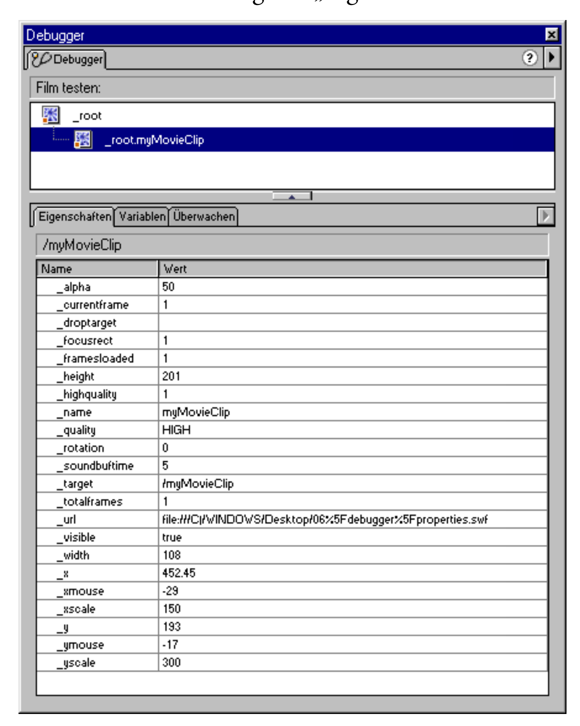
Anzeigen der Filmeigenschaften und Ändern bearbeitbarer Eigenschaften

Auf dem Register „Debuggereigenschaften“ werden sämtliche Eigenschaftswerte aller auf der Bühne befindlichen Filmsequenzen angezeigt. Die Änderung eines Eigenschaftswertes wirkt sich während der Wiedergabe auf den Film aus. Einige Eigenschaften von Filmsequenzen sind schreibgeschützt und können nicht geändert werden.

So lassen Sie sich die Eigenschaften einer Filmsequenz anzeigen:

1 Wählen Sie in der Anzeigeliste eine Filmsequenz aus.

2 Klicken Sie auf das Register „Eigenschaften“.



So ändern Sie einen Eigenschaftswert:

Markieren Sie den Wert, und geben Sie einen neuen Wert ein.

Der Wert muss eine Konstante sein (beispielsweise `50` oder `"clearwater"`) und kein Ausdruck (beispielsweise `x + 50`). Der Wert kann eine Zeichenfolge (jeder in Anführungszeichen (`""`) eingeschlossene Wert), eine Zahl oder ein Boolean (`true` oder `false`) sein. Sie können im Debugger jedoch keine Objekt- und Arraywerte (beispielsweise `{id: "rogue"}` oder `[1, 2, 3]`) eingeben.

Weitere Informationen erhalten Sie unter „Zeichenfolge“ auf Seite 56 und „Verwenden von Operatoren zum Ändern von Werten in Ausdrücken“ auf Seite 65.

Anmerkung: Im Filmtestmodus können Sie mit der Aktion `trace` den Wert eines Ausdrucks ausgeben. Siehe „Verwenden des Befehls „trace““ auf Seite 172.

Verwenden des Ausgabefensters

Im Filmtestmodus werden im Ausgabefenster Informationen angezeigt, die es Ihnen erleichtern, Fehler in Ihrem Film zu beheben. Einige Informationen wie Syntaxfehler werden automatisch angezeigt. Mit Hilfe der Befehle „Objekte auflisten“ und „Variablen auflisten“ können Sie sich weitere Informationen anzeigen lassen. (Siehe „Verwenden des Befehls „Objekte auflisten““ auf Seite 171 und „Verwenden des Befehls „Variablen auflisten““ auf Seite 171.)

Wenn Sie in Ihren Skripts die Aktion `trace` verwenden, können Sie spezifische Informationen an das Ausgabefenster senden, während der Film abgespielt wird. Dabei kann es sich um Anmerkungen zum Status des Filmes oder den Wert eines Ausdrucks handeln. Siehe „Verwenden des Befehls „trace““ auf Seite 172.

So lassen Sie sich das Ausgabefenster anzeigen:

1. Wenn der Film nicht im Filmtestmodus abgespielt wird, wählen Sie „Steuerung“ > „Film testen“.
2. Wählen Sie „Fenster“ > „Ausgabe“.

   Das Ausgabefenster wird angezeigt.

Anmerkung: Wenn das Skript Syntaxfehler enthält, wird das Ausgabefenster automatisch angezeigt.

3. Wählen Sie das Menü „Optionen“, um den Inhalt des Ausgabefensters zu bearbeiten:

- Wählen Sie „Optionen“ > „Kopieren“, um den Inhalt des Ausgabefensters in die Zwischenablage zu kopieren.
- Wählen Sie „Optionen“ > „Löschen“, um den Inhalt des Fensters zu löschen.
- Wählen Sie „Optionen“ > „In Datei speichern“, um den Inhalt des Fensters in einer Textdatei zu speichern.
- Wählen Sie „Optionen“ > „Drucken“, um den Inhalt des Fensters zu drucken.

Verwenden des Befehls „Objekte auflisten“

Mit dem Befehl „Objekte auflisten“ werden im Filmtestmodus Ebene, Bild, Objekttyp (Form, Filmsequenz oder Schaltfläche) und Zielpfad einer Filmsequenzinstanz in einer hierarchischen Liste angezeigt. Dies erleichtert insbesondere das Finden des korrekten Zielpfades und des Instanznamens. Im Gegensatz zum Debugger wird diese Liste beim Abspielen des Filmes nicht automatisch aktualisiert. Wenn Sie Informationen an das Ausgabefenster senden möchten, müssen Sie jedes Mal den Befehl „Objekte auflisten“ auswählen.

So lassen Sie sich eine Liste der Objekte in einem Film anzeigen:

1 Wenn der Film nicht im Filmtestmodus abgespielt wird, wählen Sie „Steuerung“ > „Film testen“.

2 Wählen Sie „Fehlersuche“ > „Objekte auflisten“.

Im Ausgabefenster wird eine Liste der zu diesem Zeitpunkt auf der Bühne befindlichen Objekte angezeigt, wie im folgenden Beispiel dargestellt:

```
Layer #0: Frame=3
Movie Clip: Frame=1 Target=_root.MC
    Shape:
  Movie Clip: Frame=1 Target=_root.instance3
    Shape:
  Button:
    Movie Clip: Frame=1 Target=_root.instance3.instance2
      Shape:
```

Anmerkung: Mit dem Befehl „Objekte auflisten“ werden nicht alle ActionScript-Datenobjekte aufgelistet. In diesem Zusammenhang wird eine Form oder ein Symbol auf der Bühne als Objekt bezeichnet.

Verwenden des Befehls „Variablen auflisten“

Mit dem Befehl „Variablen auflisten“ werden im Filmtestmodus alle zu diesem Zeitpunkt im Film vorhandenen Variablen angezeigt. Dies erleichtert insbesondere das Finden des korrekten Zielpfades der Variablen und des Variablennamens. Im Gegensatz zum Debugger wird diese Liste beim Abspielen des Filmes nicht automatisch aktualisiert. Wenn Sie Informationen an das Ausgabefenster senden möchten, müssen Sie jedes Mal den Befehl „Variablen auflisten“ auswählen.

So lassen Sie sich eine Liste der Variablen in einem Film anzeigen:

1 Wenn der Film nicht im Filmtestmodus abgespielt wird, wählen Sie „Steuerung“ > „Film testen“.

2 Wählen Sie „Fehlersuche“ > „Variablen auflisten“.

Im Ausgabefenster wird eine Liste der gegenwärtig im Film vorhandenen Variablen angezeigt, wie im folgenden Beispiel dargestellt:

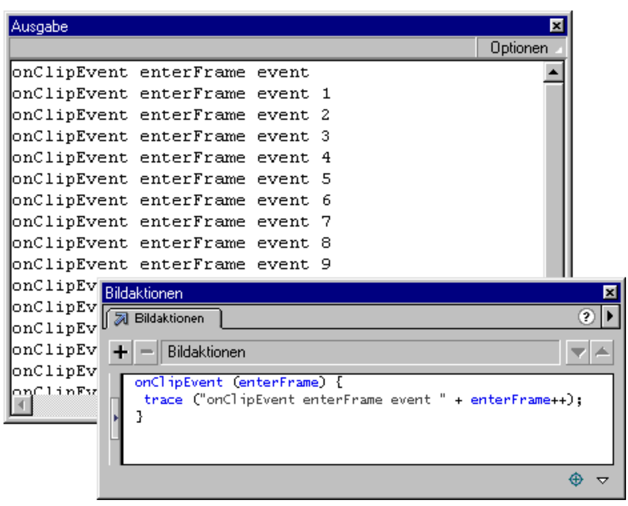
```
Level #0:
  Variable _root.country = "Schweden"
  Variable _root.city = "San Francisco"
Movie Clip:  Target=""
Variable _root.instance1.firstName = "Rick"
```

Verwenden des Befehls „trace“

Mit Hilfe der Aktion `trace` in einem Skript können Sie Informationen an das Ausgabefenster senden. Wenn Sie beispielsweise einen Film oder eine Szene testen, können Sie bestimmte Programmierhinweise an das Fenster senden. Sie können auch festlegen, dass beim Drücken einer Schaltfläche oder Abspielen eines Bildes bestimmte Ergebnisse angezeigt werden. Die Aktion `trace` ist mit der Anweisung `alert` in JavaScript vergleichbar.

Wenn Sie die Aktion `trace` in einem Skript einsetzen, können Sie Ausdrücke als Argumente verwenden. Im Filmtestmodus wird der Wert eines Ausdrucks im Ausgabefenster angezeigt, wie im folgenden Beispiel dargestellt:

```
onClipEvent(enterFrame){
   trace("onClipEvent enterFrame" + enterFrame++)
}
```



Die von der Aktion `trace` zurückgegebenen Werte werden im Ausgabefenster angezeigt.



KAPITEL 7
Referenz zu ActionScript

In diesem Teil des *ActionScript-Referenzhandbuchs* werden die Syntax und die Verwendung der ActionScript-Elemente in Flash 5 und höheren Versionen beschrieben. Die Einträge in diesem Handbuch sind identisch mit denen in der Onlinereferenz zu ActionScript. Wenn Sie Beispiele in einem Skript verwenden möchten, kopieren Sie den Beispieltext aus der Onlinereferenz zu ActionScript, und fügen Sie ihn im Expertenmodus im Bedienfeld „Aktionen" ein.

In der Referenz werden alle ActionScript-Elemente wie Operatoren, Schlüsselwörter, Anweisungen, Aktionen, Eigenschaften, Funktionen, Objekte und Methoden aufgeführt. Einen Überblick über sämtliche Einträge in der Referenz erhalten Sie unter „Inhalt der Referenz" auf Seite 176; die Tabellen in diesem Abschnitt sind gut geeignet für das Nachschlagen von symbolischen Operatoren oder Methoden, deren Objektklasse Ihnen unbekannt ist.

Sofern nicht anders angegeben, folgt ActionScript dem Standard ECMA-262, einer von der ECMA (European Computer Manufacturers Association) verfassten Spezifikation.

Diese Referenz umfasst zwei Arten von Einträgen:

- Einzelne Einträge für Operatoren, Schlüsselwörter, Funktionen, Variablen, Eigenschaften, Methoden und Anweisungen
- Objekteinträge, die allgemeine Angaben zu vordefinierten Objekten enthalten

Entnehmen Sie den Beispieleinträgen die Struktur und die Konventionen, die in diesen beiden Eintragstypen verwendet werden.

Beispieleintrag für die meisten ActionScript-Elemente

Der folgende Beispielreferenzeintrag zeigt die Konventionen, die für alle ActionScript-Elemente mit Ausnahme von Objekten verwendet werden.

Eintragstitel

Alle Einträge werden in alphabetischer Reihenfolge aufgeführt. Bei der alphabetischen Sortierung bleiben Groß- und Kleinschreibung, vorangestellte Unterstriche und Ähnliches unberücksichtigt.

Syntax

Im Abschnitt „Syntax" wird die korrekte Syntax für die Verwendung des ActionScript-Elements in Ihrem Code aufgeführt. Der Code innerhalb der Syntax wird in `Codeschriftart` angegeben, und die anzugebenden Argumente werden in *`kursiver Codeschriftart`* markiert. Klammern kennzeichnen optionale Argumente.

Argumente

In diesem Abschnitt werden alle in der Syntax aufgeführten Argumente beschrieben.

Beschreibung

In diesem Abschnitt wird das Element angegeben (zum Beispiel als Operator, Methode, Funktion oder sonstiges Element) und anschließend seine Verwendung beschrieben.

Player

In diesem Abschnitt wird angegeben, welche Versionen des Players das Element unterstützen. Dies bezeichnet nicht die Flash Version, die zur Erstellung verwendet wird. Wenn Sie z. B. mit dem Erstellungsprogramm von Flash 5 Inhalte für den Flash 4 Player erstellen, können Sie keine ActionScript-Elemente verwenden, die nur für den Flash 5 Player verfügbar sind.

Mit Einführung von Flash 5 ActionScript gelten einige ActionScript-Elemente von Flash 4 (und früheren Versionen) als veraltet. Obwohl veraltete Elemente immer noch vom Flash 5 Player unterstützt werden, wird die Verwendung der neuen Flash 5-Elemente empfohlen.

Des Weiteren wurde der Funktionsumfang der Operatoren in Flash 5 wesentlich erweitert. Es wurden eine Reihe neuer mathematischer Operatoren eingeführt, und einige ältere Operatoren sind nun in der Lage, zusätzliche Datentypen zu verarbeiten. Aus Gründen der Datentypkonsistenz werden Flash 4-Dateien automatisch angepasst, wenn sie in die Flash 5-Erstellungsumgebung importiert werden. Diese Änderungen haben jedoch keine Auswirkungen auf die Funktionsfähigkeit des ursprünglichen Skripts. Weitere Informationen finden Sie in den Einträgen zu + (Addition), < (kleiner als), > (größer als), <= (kleiner oder gleich), >= (größer oder gleich), != (nicht gleich) und = (gleich).

Beispiel
In diesem Abschnitt wird ein Codebeispiel für die Verwendung des Elements dargestellt.

Siehe auch
In diesem Abschnitt werden weiterführende Einträge in der Referenz zu ActionScript angegeben.

Beispieleintrag für Objekte

Der folgende Beispielreferenzeintrag zeigt die Konventionen, die für vordefinierte ActionScript-Objekte verwendet werden. Objekte werden mit allen anderen Elementen in alphabetischer Reihenfolge im Wörterbuch aufgeführt.

Eintragstitel

Der Eintragstitel gibt den Namen des Objektes an. Dem Objektnamen folgt ein Absatz mit allgemeinen Informationen über das Objekt.

Tabellarische Zusammenfassungen zu Methoden und Eigenschaften

Jeder Objekteintrag enthält eine Tabelle, in der alle zum Objekt gehörenden Methoden aufgeführt werden. Wenn das Objekt Eigenschaften besitzt (häufig Konstanten), werden diese Elemente in einer weiteren Tabelle zusammengefasst. Alle in diesen Tabellen aufgeführten Methoden und Eigenschaften verfügen ebenfalls über Einträge in der Referenz, die auf den Eintrag für das Objekt folgen.

Konstruktor

Wenn Sie für den Zugriff auf Methoden und Eigenschaften des Objektes einen Konstruktor verwenden müssen, wird dieser Konstruktor am Ende des Objekteintrags beschrieben. Diese Beschreibung umfasst sämtliche Standardelemente (Syntaxbeschreibung usw.) der anderen Wörterbucheinträge.

Listen der Methoden und Eigenschaften

Die Methoden und Eigenschaften eines Objektes werden in alphabetischer Reihenfolge in Anschluss an den Objekteintrag aufgeführt.

Inhalt der Referenz

Die Referenzeinträge sind in alphabetischer Reihenfolge angeordnet. Einige Operatoren sind jedoch Symbole und werden in ASCII-Reihenfolge aufgeführt. Zusätzlich werden einem Objekt zugehörige Methoden mit dem Namen des Objektes angegeben, z. B. wird die `abs`-Methode des Math-Objektes als `Math.abs` angeführt.

Die folgenden beiden Tabellen erleichtern Ihnen die Suche nach diesen Elementen. In der ersten Tabelle werden die symbolischen Operatoren in der gleichen Reihenfolge wie in der Referenz aufgeführt. In der zweiten Tabelle werden alle sonstigen ActionScript-Elemente aufgeführt.

Anmerkung: Informationen über den Vorrang und die Assoziativität der Operatoren erhalten Sie in Anhang A.

Symbolische Operatoren	
--	(Dekrement)
++	(Inkrement)
!	(logisches NICHT)
!=	(nicht gleich)
%	(Modulo)
%=	(Modulo-Zuweisung)
&	(Bitweises UND)
&&	(Kurzgeschlossenes UND)
&=	(Bitweise UND-Zuweisung)
( )	(Klammer)
-	(Minus)
*	(Multiplikation)
*=	(Multiplikationszuweisung)
,	(Komma)
.	(Punkt)
? :	(bedingt)
/	(Division)
//	(Kommentartrennzeichen)
/*	(Kommentartrennzeichen)

Symbolische Operatoren	
/=	(Divisionszuweisung)
[]	(Arrayzugriff)
^	(Bitweises XODER)
^=	(Bitweise XODER-Zuweisung)
{}	(Objektinitialisierung)
|	(Bitweises ODER)
||	(Logisches ODER)
|=	(Bitweise ODER-Zuweisung)
~	(Bitweises NICHT)
+	(Addition)
+=	(Additionszuweisung)
<	(kleiner als)
<<	(Bitweises Shift links)
<<=	(Bitweises Shift links und Zuweisung)
<=	(kleiner oder gleich)
<>	(nicht gleich)
=	(Zuweisung)
-=	(Negationszuweisung)
==	(Gleichheit)
>	(größer als)
>=	(größer oder gleich)
>>	(Bitweises Shift rechts)
>>=	(Bitweises Shift rechts und Zuweisung)
>>>	(Vorzeichenloses bitweises Shift rechts)
>>>=	(Vorzeichenloses bitweises Shift rechts und Zuweisung)

In der folgenden Tabelle werden alle ActionScript-Elemente aufgelistet, die keine symbolischen Operatoren darstellen.

ActionScript-Element	Siehe Eintrag
abs	„Math.abs" auf Seite 296
acos	„Math.acos" auf Seite 296
add	„add" auf Seite 221
and	„and" auf Seite 222
_alpha	„_alpha" auf Seite 221
appendChild	„XML.appendChild" auf Seite 391
Array	„Array (Objekt)" auf Seite 222
asin	„Math.asin" auf Seite 297
atan	„Math.atan" auf Seite 297
atan2	„Math.atan2" auf Seite 297
attachMovie	„MovieClip.attachMovie" auf Seite 311
attachSound	„Sound.attachSound" auf Seite 355
attributes	„XML.attributes" auf Seite 392
RÜCKSCHRITTTASTE	„Key.BACKSPACE" auf Seite 281
Boolean	„Boolean (Funktion)" auf Seite 232, „Boolean (Objekt)" auf Seite 232
break	„break" auf Seite 234
call	„call" auf Seite 235
FESTSTELLTASTE	„Key.CAPSLOCK" auf Seite 282
ceil	„Math.ceil" auf Seite 298
charAt	„String.charAt" auf Seite 368
charCodeAt	„String.charCodeAt" auf Seite 368
childNodes	„XML.childNodes" auf Seite 392
chr	„chr" auf Seite 235
cloneNode	„XML.cloneNode" auf Seite 393
close	„XMLSocket.close" auf Seite 407
Color	„Color (Objekt)" auf Seite 236

ActionScript-Element	Siehe Eintrag
concat	„Array.concat“ auf Seite 224, „String.concat“ auf Seite 369
connect	„XMLSocket.connect“ auf Seite 408
Konstruktor	Array, Boolean, Color, Date, Number, Object, Sound, String, XML, XMLSocket
continue	„continue“ auf Seite 239
STRG	„Key.CONTROL“ auf Seite 282
cos	„Math.cos“ auf Seite 298
createElement	„XML.createElement“ auf Seite 393
createTextNode	„XML.createTextNode“ auf Seite 393
_currentframe	„_currentframe“ auf Seite 240
Date	„Date (Objekt)“ auf Seite 241
delete	„delete“ auf Seite 258
ENTF	„Key.DELETEKEY“ auf Seite 282
docTypeDecl	„XML.docTypeDecl“ auf Seite 394
do...while	„do... while“ auf Seite 260
NACH UNTEN	„Key.DOWN“ auf Seite 283
_droptarget	„_droptarget“ auf Seite 261
duplicateMovieClip	„duplicateMovieClip“ auf Seite 262, „MovieClip.duplicateMovieClip“ auf Seite 312
E	„Math.E“ auf Seite 299
else	„else {“ auf Seite 263
ENDE	„Key.END“ auf Seite 283
EINGABETASTE	„Key.ENTER“ auf Seite 283
eq	„eq (gleich - zeichenfolgenspezifisch)“ auf Seite 263
escape (Funktion)	„escape“ auf Seite 263
ESC (Konstante)	„Key.ESCAPE“ auf Seite 284
eval	„eval“ auf Seite 264
evaluate	„evaluate“ auf Seite 265
exp	„Math.exp“ auf Seite 299

ActionScript-Element	Siehe Eintrag
firstChild	„XML.firstChild“ auf Seite 395
floor	„Math.floor“ auf Seite 300
_focusrect	„_focusrect“ auf Seite 265
for	„for“ auf Seite 266
for.. in	„for..in“ auf Seite 267
_framesloaded	„_framesloaded“ auf Seite 268
fromCharCode	„String.fromCharCode“ auf Seite 369
fscommand	„fscommand“ auf Seite 269
function	„function“ auf Seite 269
ge	„ge (größer oder gleich - zeichenfolgenspezifisch)“ auf Seite 271
getAscii	„Key.getAscii“ auf Seite 284
getBeginIndex	„Selection.getBeginIndex“ auf Seite 349
getBounds	„MovieClip.getBounds“ auf Seite 313
getBytesLoaded	„MovieClip.getBytesLoaded“ auf Seite 313
getBytesTotal	„MovieClip.getBytesTotal“ auf Seite 314
getCaretIndex	„Selection.getCaretIndex“ auf Seite 350
getCode	„Key.getCode“ auf Seite 284
getDate	„Date.getDate“ auf Seite 245
getDay	„Date.getDay“ auf Seite 245
getEndIndex	„Selection.getEndIndex“ auf Seite 350
getFocus	„Selection.getFocus“ auf Seite 350
getFullYear	„Date.getFullYear“ auf Seite 245
getHours	„Date.getHours“ auf Seite 246
getMilliseconds	„Date.getMilliseconds“ auf Seite 246
getMinutes	„Date.getMinutes“ auf Seite 246
getMonth	„Date.getMonth“ auf Seite 247
getPan	„Sound.getPan“ auf Seite 355
getProperty	„getProperty“ auf Seite 271

ActionScript-Element	Siehe Eintrag
getRGB	„Color.setRGB“ auf Seite 237
getSeconds	„Date.getSeconds“ auf Seite 247
getTime	„Date.getTime“ auf Seite 247
getTimer	„getTimer“ auf Seite 272
getTimezoneOffset	„Date.getTimezoneOffset“ auf Seite 248
getTransform	„Color.getTransform“ auf Seite 237, „Sound.getTransform“ auf Seite 356
getURL	„getURL“ auf Seite 272, „MovieClip.getURL“ auf Seite 314
getUTCDate	„Date.getUTCDate“ auf Seite 248
getUTCDay	„Date.getUTCDay“ auf Seite 249
getUTCFullYear	„Date.getUTCFullYear“ auf Seite 249
getUTCHours	„Date.getUTCHours“ auf Seite 249
getUTCMilliseconds	„Date.getUTCMilliseconds“ auf Seite 249
getUTCMinutes	„Date.getUTCMinutes“ auf Seite 250
getUTCMonth	„Date.getUTCMonth“ auf Seite 250
getUTCSeconds	„Date.getUTCSeconds“ auf Seite 250
getVersion	„getVersion“ auf Seite 273
getVolume	„Sound.getVolume“ auf Seite 356
getYear	„Date.getYear“ auf Seite 251
globalToLocal	„MovieClip.globalToLocal“ auf Seite 315
gotoAndPlay	„gotoAndPlay“ auf Seite 274, „MovieClip.gotoAndPlay“ auf Seite 315
gotoAndStop	„gotoAndStop“ auf Seite 274, „MovieClip.gotoAndStop“ auf Seite 316
gt	„gt (größer - zeichenfolgenspezifisch)“ auf Seite 275
hasChildNodes	„XML.haschildNodes“ auf Seite 395
_height	„_height“ auf Seite 275
hide	„Mouse.hide“ auf Seite 309
_highquality	„_highquality“ auf Seite 276

ActionScript-Element	Siehe Eintrag
hitTest	„MovieClip.hitTest“ auf Seite 316
POS1	„Key.HOME“ auf Seite 285
if	„if“ auf Seite 276
ifFrameLoaded	„ifFrameLoaded“ auf Seite 277
#include	„#include“ auf Seite 278
indexOf	„String.indexOf“ auf Seite 369
Infinity	„Infinity“ auf Seite 278
EINFG	„Key.INSERT“ auf Seite 285
insertBefore	„XML.insertBefore“ auf Seite 396
int	„int“ auf Seite 278
isDown	„Key.isDown“ auf Seite 285
isFinite	„isFinite“ auf Seite 279
isNaN	„isNaN“ auf Seite 279
isToggled	„Key.isToggled“ auf Seite 286
join	„Array.join“ auf Seite 225
Key	„Key (Objekt)“ auf Seite 280
lastChild	„XML.lastChild“ auf Seite 396
lastIndexOf	„String.indexOf“ auf Seite 369
le	„le (kleiner oder gleich - zeichenfolgenspezifisch)“ auf Seite 289
NACH LINKS	„Key.LEFT“ auf Seite 286
length	„length“ auf Seite 289, „Array.length“ auf Seite 226, „String.length“ auf Seite 370
LN2	„Math.LN2“ auf Seite 301
LN10	„Math.LN10“ auf Seite 302
load	„XML.load“ auf Seite 397
loaded	„XML.loaded“ auf Seite 397
loadMovie	„loadMovie“ auf Seite 291, „MovieClip.loadMovie“ auf Seite 317

ActionScript-Element	Siehe Eintrag
loadVariables	„loadVariables“ auf Seite 292, „MovieClip.loadVariables“ auf Seite 318
localToGlobal	„MovieClip.localToGlobal“ auf Seite 319
log	„Math.log“ auf Seite 300
LOG2E	„Math.LOG2E“ auf Seite 300
LOG10E	„Math.LOG10E“ auf Seite 301
lt	„le (kleiner oder gleich - zeichenfolgenspezifisch)“ auf Seite 289
Math	„Math (Objekt)“ auf Seite 294
max	„Math.max“ auf Seite 302
maxscroll	„maxscroll“ auf Seite 306
MAX_VALUE	„Number.MAX_VALUE“ auf Seite 330
mbchr	„mbchr“ auf Seite 307
mblength	„mblength“ auf Seite 307
mbord	„mbord“ auf Seite 308
mbsubstring	„mbsubstring“ auf Seite 308
min	„Math.min“ auf Seite 302
MIN_VALUE	„Number.MIN_VALUE“ auf Seite 330
Mouse	„Mouse (Objekt)“ auf Seite 308
MovieClip	„MovieClip (Objekt)“ auf Seite 310
_name	„_name“ auf Seite 323
NaN	„NaN“ auf Seite 323, „Number.NaN“ auf Seite 330
ne	„ne (ungleich – zeichenfolgenspezifisch)“ auf Seite 324
NEGATIVE_INFINITY	„Number.NEGATIVE_INFINITY“ auf Seite 331
new **(Operator)**	„new“ auf Seite 324
newline	„newline“ auf Seite 325
nextFrame	„nextFrame“ auf Seite 325, „MovieClip.nextFrame“ auf Seite 319
nextScene	„nextScene“ auf Seite 326
nextSibling	„XML.nextSibling“ auf Seite 398

ActionScript-Element	Siehe Eintrag
nodeName	„XML.nodeName“ auf Seite 398
nodeType	„XML.nodeType“ auf Seite 399
nodeValue	„XML.nodeValue“ auf Seite 399
not	„not“ auf Seite 326
null	„null“ auf Seite 327
Number	„Number (Funktion)“ auf Seite 327, „Number (Objekt)“ auf Seite 328
Object	„Object (Objekt)“ auf Seite 332
On	„on(mouseEvent)“ auf Seite 336
onClipEvent	„onClipEvent“ auf Seite 334
onClose	„XMLSocket.onClose“ auf Seite 409
onConnect	„XMLSocket.onConnect“ auf Seite 410
OnLoad	„XML.onLoad“ auf Seite 400
onXML	„XMLSocket.onXML“ auf Seite 411
or (logisches ODER)	„or“ auf Seite 337
ord	„ord“ auf Seite 338
_parent	„_parent“ auf Seite 338
parentNode	„XML.parentNode“ auf Seite 401
parseFloat	„parseFloat“ auf Seite 339
parseInt	„parseInt“ auf Seite 339
parseXML	„XML.parseXML“ auf Seite 401
BILD-AB	„Key.PGDN“ auf Seite 286
BILD-AUF	„Key.PGUP“ auf Seite 287
PI	„Math.PI“ auf Seite 303
play	„play“ auf Seite 340, „MovieClip.play“ auf Seite 320
pop	„Array.pop“ auf Seite 227
POSITIVE_INFINITY	„Number.POSITIVE_INFINITY“ auf Seite 331
pow	„Math.pow“ auf Seite 303

ActionScript-Element	Siehe Eintrag
prevFrame	„prevFrame“ auf Seite 341, „MovieClip.prevFrame“ auf Seite 320
previousSibling	„XML.previousSibling“ auf Seite 401
prevScene	„prevScene“ auf Seite 341
print	„print“ auf Seite 342
printAsBitmap	„printAsBitmap“ auf Seite 343
push	„Array.push“ auf Seite 227
_quality	„_quality“ auf Seite 344
random	„random“ auf Seite 345, „Math.random“ auf Seite 304
removeMovieClip	„removeMovieClip“ auf Seite 346, „MovieClip.removeMovieClip“ auf Seite 320
removeNode	„XML.removeNode“ auf Seite 402
return	„return“ auf Seite 346
reverse	„Array.reverse“ auf Seite 228
NACH RECHTS	„Key.RIGHT“ auf Seite 287
_root	„_root“ auf Seite 347
_rotation	„_rotation“ auf Seite 348
round	„Math.round“ auf Seite 304
scroll	„scroll“ auf Seite 348
Selection	„Selection (Objekt)“ auf Seite 349
send	„XML.send“ auf Seite 402, „XMLSocket.send“ auf Seite 412
sendAndLoad	„XML.sendAndLoad“ auf Seite 403
set	„set“ auf Seite 351
setDate	„Date.setDate“ auf Seite 251
setFocus	„Selection.setFocus“ auf Seite 351
setFullYear	„Date.setFullYear“ auf Seite 251
setHours	„Date.setHours“ auf Seite 252
setMilliseconds	„Date.setMilliseconds“ auf Seite 252
setMinutes	„Date.setMinutes“ auf Seite 253

ActionScript-Element	Siehe Eintrag
setMonth	„Date.setMonth“ auf Seite 253
setPan	„Sound.setPan“ auf Seite 356
setProperty	„setProperty“ auf Seite 353
setRGB	„Color.setRGB“ auf Seite 237
setSeconds	„Date.setSeconds“ auf Seite 253
setSelection	„Selection.setSelection“ auf Seite 351
setTime	„Date.setTime“ auf Seite 254
setTransform	„Color.setTransform“ auf Seite 238, „Sound.setTransform“ auf Seite 357
setUTCDate	„Date.setUTCDate“ auf Seite 254
setUTCFullYear	„Date.setUTCFullYear“ auf Seite 254
setUTCHours	„Date.setUTCHours“ auf Seite 255
setUTCMilliseconds	„Date.setUTCMilliseconds“ auf Seite 255
setUTCMinutes	„Date.setUTCMinutes“ auf Seite 256
setUTCMonth	„Date.setUTCMonth“ auf Seite 256
setUTCSeconds	„Date.setUTCSeconds“ auf Seite 256
setVolume	„Sound.setVolume“ auf Seite 360
setYear	„Date.setYear“ auf Seite 257
shift (Methode)	„Array.shift“ auf Seite 228
Umschalt (Konstante)	„Key.SHIFT“ auf Seite 287
show	„Mouse.show“ auf Seite 309
sin	„Math.sin“ auf Seite 304
slice	„Array.slice“ auf Seite 229, „String.slice“ auf Seite 371
sort	„Array.sort“ auf Seite 229
Sound	„Sound (Objekt)“ auf Seite 353
_soundbuftime	„_soundbuftime“ auf Seite 362
LEERTASTE	„Key.SPACE“ auf Seite 288
splice	„Array.splice“ auf Seite 231
split	„String.split“ auf Seite 371

ActionScript-Element	Siehe Eintrag
sqrt	„Math.sqrt“ auf Seite 305
SQRT1_2	„Math.SQRT1_2“ auf Seite 305
SQRT2	„Math.SQRT2“ auf Seite 306
start	„Sound.start“ auf Seite 361
startDrag	„startDrag“ auf Seite 362, „MovieClip.startDrag“ auf Seite 321
status	„XML.status“ auf Seite 403
stop	„stop“ auf Seite 363,„MovieClip.stop“ auf Seite 321,„Sound.stop“ auf Seite 361
stopAllSounds	„stopAllSounds“ auf Seite 363
stopDrag	„stopDrag“ auf Seite 364, „MovieClip.stopDrag“ auf Seite 321
String	„String (Funktion)“ auf Seite 364, „String (Objekt)“ auf Seite 366,„" " (Zeichenfolgentrennzeichen)“ auf Seite 365
substr	„String.substr“ auf Seite 372
substring	„substring“ auf Seite 373, „String.substring“ auf Seite 372
swapDepths	„MovieClip.swapDepths“ auf Seite 322
TAB	„Key.TAB“ auf Seite 288
tan	„Math.tan“ auf Seite 306
_target	„_target“ auf Seite 374
targetPath	„targetPath“ auf Seite 374
tellTarget	„tellTarget“ auf Seite 375
this	„this“ auf Seite 376
toggleHighQuality	„toggleHighQuality“ auf Seite 377
toLowerCase	„String.toLowerCase“ auf Seite 373
toString	„Array.toString“ auf Seite 231,„Boolean.toString“ auf Seite 234, „Date.toString“ auf Seite 257, „Number.toString“ auf Seite 332, „Object.toString“ auf Seite 333, „XML.toString“ auf Seite 404
_totalframes	„_totalframes“ auf Seite 377
toUpperCase	„String.toUpperCase“ auf Seite 373

ActionScript-Element	Siehe Eintrag
trace	„trace“ auf Seite 378
typeof	„typeof“ auf Seite 379
unescape	„unescape“ auf Seite 379
unloadMovie	„unloadMovie“ auf Seite 380, „MovieClip.unloadMovie“ auf Seite 323
unshift	„Array.shift“ auf Seite 228
NACH OBEN	„Key.UP“ auf Seite 288
updateAfterEvent	„updateAfterEvent“ auf Seite 380
_url	„_url“ auf Seite 381
UTC	„Date.UTC“ auf Seite 258
valueOf	„Boolean.valueOf“ auf Seite 234, „Number.valueOf“ auf Seite 332, „Object.valueOf“ auf Seite 334
var	„var“ auf Seite 381
_visible	„_visible“ auf Seite 382
void	„void“ auf Seite 382
while	„while“ auf Seite 383
_width	„_width“ auf Seite 384
with	„with“ auf Seite 385
_x	„_x“ auf Seite 387
XML	„XML (Objekt)“ auf Seite 388
xmlDecl	„XML.xmlDecl“ auf Seite 405
XMLSocket	„XMLSocket (Objekt)“ auf Seite 405
_xmouse	„_xmouse“ auf Seite 413
_xscale	„_xscale“ auf Seite 413
_y	„_y“ auf Seite 414
_ymouse	„_ymouse“ auf Seite 414
_yscale	„_yscale“ auf Seite 415

-- (Dekrement)

Syntax

--ausdruck
ausdruck--

Argumente

ausdruck Eine Variable, eine Zahl, ein Element in einem Array oder eine Objekteigenschaft.

Beschreibung

Operator; ein unärer Prä-Dekrement- und Post-Dekrement-Operator, der von *ausdruck* 1 subtrahiert. Die Prä-Dekrement-Form des Operators (*--ausdruck*) subtrahiert 1 von *ausdruck* und gibt das Ergebnis zurück. Die Post-Dekrement-Form des Operators (*ausdruck--*) subtrahiert 1 von *ausdruck* und gibt den Ausgangswert von *ausdruck* (das Ergebnis vor der Subtraktion) zurück.

Player

Flash 4 oder höher.

Beispiel

Die Prä-Dekrement-Form des Operators dekrementiert `x` zu 2 (`x - 1 = 2`) und gibt das Ergebnis als `y` zurück:

```
x = 3;
y = --x
```

Die Post-Dekrement-Form des Operators dekrementiert `x` zu 2 (`x - 1 = 2`) und gibt den Ausgangswert (`x = 3`) als Ergebnis `y` zurück:

If `x = 3;`

```
y = x--
```

++ (Inkrement)

Syntax

++*ausdruck*
ausdruck++

Argumente

ausdruck Eine Variable, eine Zahl, ein Element in einem Array oder eine Objekteigenschaft.

Beschreibung

Operator; ein unärer Prä-Inkrement- und Post-Inkrement-Operator, der zu *Ausdruck* 1 addiert. Die Prä-Inkrement-Form des Operators (++*ausdruck*) addiert 1 zu *ausdruck* und gibt das Ergebnis zurück. Die Post-Inkrement-Form des Operators (*ausdruck*++) addiert 1 zu *ausdruck* und gibt den Ausgangswert von *ausdruck* (das Ergebnis vor der Addition) zurück.

Die Prä-Inkrement-Form des Operators inkrementiert `x` zu 2 (`x + 1 = 2`) und gibt das Ergebnis als `y` zurück:

```
x = 1;
y = ++x
```

Die Post-Inkrement-Form des Operators inkrementiert `x` zu 2 (`x + 1 = 2`) und gibt den Ausgangswert (`x = 1`) als Ergebnis `y` zurück:

```
x = 1;
y = x++
```

Player

Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird ++ als Prä-Inkrement-Operator mit einer `while`-Anweisung verwendet.

```
i = 0
while(i++ < 5){
// Dieser Abschnitt wird fünfmal ausgeführt
}
```

Im folgenden Beispiel wird ++ als Prä-Inkrement-Operator verwendet:

```
var a = [];
var i = 0;
while (i < 10) {
   a.push(++i);
}
trace(a.join());
```

Die Ausgabe dieses Skripts lautet wie folgt:

```
1,2,3,4,5,6,7,8,9
```

Im folgenden Beispiel wird ++ als Post-Inkrement-Operator verwendet:

```
var a = [];
var i = 0;
while (i < 10) {
a.push(i++);
   }
trace(a.join());
```

Die Ausgabe dieses Skripts lautet wie folgt:

```
0,1,2,3,4,5,6,7,8,9
```

! (logisches NICHT)

Syntax
`!ausdruck`

Argumente
ausdruck Eine Variable oder ein ausgewerteter Ausdruck.

Beschreibung
Operator (logisch); kehrt den Booleschen Wert einer Variable oder eines Ausdrucks um. Wenn *ausdruck* eine Variable mit einem absoluten oder umgewandelten Wert `true` ist, ist `!variable` als Wert von `! ausdruck` gleich `false`. Wenn der Ausdruck `x && y` den Wert `false` ergibt, ergibt der Ausdruck `!(x && y)` den Wert `true`. Dieser Operator ist identisch mit dem `not`-Operator aus Flash 4.

Player
Flash 4 oder höher.

Beispiel
Im folgenden Beispiel wird die Variable `happy` auf `false` gesetzt, und die `if`-Bedingung wertet die Bedingung `!happy` aus. Wenn die Bedingung `true` ist, sendet `trace` eine Zeichenfolge an das Ausgabefenster.

```
happy = false;
if (!happy){
trace("don't worry be happy");
}
```

Im Folgenden werden die Ergebnisse des `!`-Operators demonstriert:

`! true` gibt `false` zurück

`! false` gibt `true` zurück

!= (nicht gleich)

Syntax
`ausdruck1 != ausdruck2`

Argumente
ausdruck1, ausdruck2 Zahlen, Zeichenfolgen, Booleans, Variablen, Objekte, Arrays oder Funktionen.

Beschreibung
Operator (Gleichheit); prüft auf das genaue Gegenteil des ==-Operators. Wenn *ausdruck1* gleich *ausdruck2* ist, ist das Ergebnis `false`. Wie beim ==-Operator hängt die Definition von *gleich* von den verglichenen Datentypen ab.

- Zahlen, Zeichenfolgen und Boolesche Werte werden anhand ihres Werts verglichen.
- Variablen, Objekte, Arrays und Funktionen werden anhand ihrer Referenz verglichen.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel werden die Ergebnisse des `!=`-Operators demonstriert.

`5 != 8` gibt `true` zurück

`5 != 5` gibt `false` zurück

Im folgenden Beispiel wird die Verwendung des `!=`-Operators in einer `if`-Anweisung demonstriert:

```
a = "David";
b = "Dummkopf"
if (a != b)
trace("David ist kein Dummkopf");
```

Siehe auch
„== (gleich)“ auf Seite 215

% (Modulo)

Syntax
`ausdruck1 % ausdruck2`

Argumente
`ausdruck1, ausdruck2` Zahlen, Ganzzahlen, Gleitkommazahlen oder Zeichenfolgen, die in einen numerischen Wert umgewandelt werden können.

Beschreibung
Operator (arithmetisch); berechnet den Rest einer Division von *`ausdruck1`* durch *`ausdruck2`*. Wenn alle Argumente *`ausdruck`* nicht numerisch sind, versucht der Modulo-Operator sie in Zahlen umzuwandeln.

Player
Flash 4 oder höher. In Flash 4 Dateien wird der `%`-Operator in der SWF-Datei zu `x - int(x/y) * y` erweitert und ist unter Umständen nicht so genau und schnell wie die Implementierung im Flash 5 Player.

Beispiel
Das folgende Beispiel demonstriert die numerische Verwendung des `%`-Operators:

`12 % 5` gibt `2` zurück

`4.3 % 2.1` gibt `0.1` zurück

%= (Modulo-Zuweisung)

Syntax
ausdruck1 `%=` *ausdruck2*

Argumente
ausdruck1,ausdruck2 Ganzzahlen und Variablen.

Beschreibung
Operator (Zuweisung); weist *ausdruck1* den Wert von *ausdruck1* `%` *ausdruck2* zu.

Player
Flash 4 oder höher.

Beispiel
Das folgende Beispiel demonstriert die Verwendung des `%=`-Operators mit Variablen und Zahlen:

`x %= y` ist gleich `x = x % y`

Wenn `x = 14` und `y = 5`, dann

`x %= 5` gibt `4` zurück

Siehe auch
„`% (Modulo)`“ auf Seite 192

& (Bitweises UND)

Syntax
ausdruck1 `&` *ausdruck2*

Argumente
ausdruck1, ausdruck2 Beliebige Zahl.

Beschreibung
Operator (bitweise); wandelt *ausdruck1* und *ausdruck2* in vorzeichenlose 32-Bit-Ganzzahlen um und führt eine Boolean-UND-Operation für jedes einzelne Bit der ganzzahligen Argumente aus. Das Ergebnis ist eine neue vorzeichenlose 32-Bit-Ganzzahl.

Player
Flash 5 oder höher. In Flash 4 wurde der `&`-Operator zum Verketten von Zeichenfolgen verwendet. In Flash 5 ist der `&`-Operator ein bitweises UND. Mit den Operatoren `add` und `+` werden Zeichenfolgen verkettet. Flash 4 Dateien, die den `&`-Operator verwenden, werden automatisch auf die Verwendung von `add` aktualisiert, wenn sie in der Flash 5 Erstellungsumgebung verwendet werden.

&& (kurzgeschlossenes UND)

Syntax

ausdruck1 && *ausdruck2*

Argumente

ausdruck1, ausdruck2 Zahlen, Zeichenfolgen, Variablen oder Funktionen.

Beschreibung

Operator (logisch); führt für die Werte eines oder beider Ausdrücke eine Boolesche Operation durch. Weist den Flash Interpreter an, *ausdruck1* (den linken Ausdruck) auszuwerten und gibt `false` zurück, wenn der Ausdruck den Wert `false` ergibt. Wenn *ausdruck1* den Wert `true` ergibt, wird *ausdruck2* (der rechte Ausdruck) ausgewertet. Wenn *ausdruck2* den Wert `true` ergibt, ist das Endergebnis `true`; andernfalls ist das Endergebnis `false`.

Player

Flash 4 oder höher.

Beispiel

In diesem Beispiel werden die Werte der ausgewerteten Ausdrücke den Variablen `winner` und `loser` für eine folgende Prüfung zugewiesen:

```
winner = (chocolateEggs >=10) && (jellyBeans >=25);
loser = (chocolateEggs <=1) && (jellyBeans <= 5);
if (winner) {
   alert = "Sie haben gewonnen!";
   if (loser) {
      alert = "Das nennt sich wohl Waidmannspech!";
   }
} else {
   alert = "Es gibt keine Verlierer!";
}
```

&= (bitweise UND-Zuweisung)

Syntax

ausdruck1 &= *ausdruck2*

Argumente

ausdruck1, ausdruck2 Ganzzahlen und Variablen.

Beschreibung

Operator (bitweise Zuweisung); weist *ausdruck1* den Wert von *ausdruck1* & *ausdruck2* zu.

Player

Flash 5 oder höher.

Beispiel

Das folgende Beispiel demonstriert die Verwendung des `&=`-Operators bei Variablen und Zahlen:

`x &= y` ist gleich `x = x & y`

Wenn `x = 15` und `y = 9`, dann gilt:

`x &= 9` gibt `9` zurück

Siehe auch

„& (Bitweises UND)“ auf Seite 193

() (Klammer)

Syntax

```
(ausdruck1, ausdruck2);
```

```
funktion(funktionsaufruf1, ..., funktionsaufrufN);
```

Argumente

`ausdruck1, ausdruck2` Zahlen, Zeichenfolgen, Variablen oder Text.

`funktion` Die Funktion, die mit dem Inhalt der Klammer ausgeführt werden soll.

`funktionsaufruf1...funktionsaufrufN` Eine Folge von Funktionen, die ausgeführt werden, bevor das Ergebnis an die Funktion außerhalb der Klammer übergeben wird.

Beschreibung

Operator (allgemein); führt eine Gruppierungsoperation mit einem oder mehreren Argumenten durch, oder umgibt ein oder mehrere Argumente und übergibt die Ergebnisse als Parameter an eine Funktion außerhalb der Klammer.

Verwendung 1: Führt eine Gruppierungsoperation mit einem oder mehreren Ausdrücken durch, um die Reihenfolge zu steuern, in der die Operatoren im Ausdruck ausgeführt werden. Dieser Operator unterdrückt die automatische Vorrangreihenfolge und erzwingt eine vorrangige Auswertung der Ausdrücke innerhalb der Klammer. Bei verschachtelten Klammern wertet Flash den Inhalt der innersten Klammer vor dem Inhalt der äußeren aus.

Verwendung 2: Umgibt ein oder mehrere Argumente und gibt diese als Parameter an die Funktion außerhalb der Klammer weiter.

Player

Flash 4 oder höher.

Beispiel

(Verwendung 1) Die folgenden Anweisungen demonstrieren die Verwendung von Klammern zur Steuerung der Ausführungsreihenfolge von Ausdrücken. (Das Ergebnis wird unter den einzelnen Anweisungen abgedruckt.)

```
(2 + 3) * (4 + 5)
45
2 + (3 * (4 + 5))
29
2 + (3 * 4) + 5
19
```

(Verwendung 2) Das folgende Beispiel demonstriert die Verwendung von Klammern mit einer Funktion:

```
getDate();
invoice(item, amount);
```

Siehe auch

„with“ auf Seite 385

- (Minus)

Syntax

(Negation) –*ausdruck*

(Subtraktion) *ausdruck1* - *ausdruck2*

Argumente

ausdruck1, ausdruck2 Beliebige Zahl.

Beschreibung

Operator (arithmetisch); dient zur Negation oder Subtraktion. Als Negation kehrt er das Vorzeichen des numerischen Ausdrucks *ausdruck* um. Als Subtraktion wird eine arithmetische Subtraktion von zwei numerischen Ausdrücken durchgeführt, wobei *ausdruck2* von *ausdruck1* subtrahiert wird. Wenn beide Ausdrücke Ganzzahlen sind, ist die Differenz eine Ganzzahl. Wenn einer oder beide Ausdrücke Gleitkommazahlen sind, ist die Differenz eine Gleitkommazahl.

Player

Flash 4 oder höher.

Beispiel

(Negation) Diese Anweisung kehrt das Vorzeichen des Ausdrucks 2 + 3 um:

```
-(2 + 3)
```

Das Ergebnis ist -5.

(Subtraktion) Diese Anweisung subtrahiert die Ganzzahl 2 von der Ganzzahl 5:

```
5 - 2
```

Das Ergebnis ist die Ganzzahl 3.

(Subtraktion): Diese Anweisung subtrahiert die Gleitkommazahl 1,5 von der Gleitkommazahl 3,25:

```
put 3.25 - 1.5
```

Das Ergebnis ist die Gleitkommazahl 1,75.

* (Multiplikation)

Syntax

*ausdruck1 * ausdruck2*

Argumente

ausdruck1, ausdruck2 Ganzzahlen oder Gleitkommazahlen.

Beschreibung

Operator (arithmetisch); multipliziert zwei numerische Ausdrücke. Wenn beide Ausdrücke Ganzzahlen sind, ist das Produkt eine Ganzzahl. Wenn einer oder beide Ausdrücke Gleitkommazahlen sind, ist das Produkt eine Gleitkommazahl.

Player

Flash 4 oder höher.

Beispiel

Diese Anweisung multipliziert die Ganzzahlen 2 und 3:

```
2 * 3
```

Das Ergebnis ist die Ganzzahl 6.

Beispiel

Diese Anweisung multipliziert die Gleitkommazahlen 2,0 und 3,1416:

```
2.0 * 3.1416
```

Das Ergebnis ist die Gleitkommazahl 6,2832.

*= (Multiplikationszuweisung)

Syntax

*ausdruck1 *= ausdruck2*

Argumente

ausdruck1, ausdruck2 Ganzzahlen, Gleitkommazahlen oder Zeichenfolgen.

Beschreibung

Operator (Zuweisung); weist *ausdruck1* den Wert von *ausdruck1 * ausdruck2* zu.

Player

Flash 4 oder höher.

Beispiel
Das folgende Beispiel demonstriert die Verwendung des `*=`-Operators bei Variablen und Zahlen:

`x *= y` ist gleich `x = x * y`

Wenn `x = 5` und `y = 10`, dann gilt:

`x *= 10` gibt `50` zurück

Siehe auch
„* (Multiplikation)“ auf Seite 197

, (Komma)

Syntax
ausdruck1, *ausdruck2*

Argumente
ausdruck Beliebige Zahl, Variable, Zeichenfolge, Arrayelement oder sonstige Daten.

Beschreibung
Operator; weist Flash an, *ausdruck1* und anschließend *ausdruck2* auszuwerten und anschließend den Wert von *ausdruck2* zurückzugeben. Dieser Operator wird hauptsächlich mit der `for`-Schleifenanweisung verwendet.

Player
Flash 4 oder höher.

Beispiel
Im folgenden Codebeispiel wird der Komma-Operator verwendet:

```
var a=1, b=2, c=3;
```

Dies ist gleichbedeutend mit folgender Schreibweise:

```
var a=1;
var b=2;
var c=3;
```

. (Punkt-Operator)

Syntax
objekt.eigenschaft_oder_methode
instanzname.variable
instanzname.unterinstanz.variable

Argumente

objekt Die Instanz eines Objektes. Bei einigen Objekten müssen Instanzen mit Hilfe des Konstruktors erstellt werden. Bei dem Objekt kann es sich um ein beliebiges vordefiniertes ActionScript-Objekt oder ein benutzerdefiniertes Objekt handeln. Dieses Argument steht immer links vom Punkt-Operator (.).

eigenschaft_oder_methode Der Name einer Eigenschaft oder Methode, die einem Objekt zugeordnet ist. Alle gültigen Methoden und Eigenschaften der vordefinierten Objekte sind in den tabellarischen Zusammenfassungen der Methoden und Eigenschaften für das entsprechende Objekt aufgeführt. Dieses Argument steht immer rechts vom Punkt-Operator (.).

instanzname Der Name der Instanz einer Filmsequenz.

unterinstanz Die Instanz einer Filmsequenz, die eine untergeordnete Filmsequenz der Hauptfilmsequenz darstellt.

variable Eine Variable in einer Filmsequenz.

Beschreibung

Operator; wird zur Navigation in Hierarchien von Filmsequenzen verwendet, um auf verschachtelte untergeordnete Filmsequenzen, Variablen oder Eigenschaften zuzugreifen. Der Punkt-Operator wird auch zum Testen oder Festlegen der Eigenschaften eines Objektes, zum Ausführen einer Objektmethode oder zum Erstellen einer Datenstruktur verwendet.

Player

Flash 4 oder höher.

Siehe auch

„[] (Arrayzugriffsoperator)“ auf Seite 202

Beispiel

Mit dieser Anweisung wird der aktuelle Wert der Variablen `hairColor` in der Filmsequenz `person` ermittelt:

```
person.hairColor
```

Dies entspricht der folgenden Syntax in Flash 4:

```
/person:hairColor
```

Beispiel

Der folgende Code demonstriert die Verwendung des Punkt-Operators zum Erstellen einer Arraystruktur:

```
account.name = "Georg Schmidt";
account.address = "Hauptstr. 123";
account.city = "Astadt";
account.state = "D";
account.zip = "12345";
```

?: (bedingt)

Syntax

ausdruck1 ? *ausdruck2* : *ausdruck3*

Argumente

ausdruck1 Ein Ausdruck, der einen Booleschen Wert annimmt, in der Regel ein Vergleichsausdruck.

ausdruck2, ausdruck3 Werte von einem beliebigen Typ.

Beschreibung

Operator (bedingt); weist Flash an, *ausdruck1* auszuwerten und den Wert von *ausdruck2* zurückzugeben, wenn *ausdruck1* den Wert `true` hat; andernfalls wird der Wert von *ausdruck3* zurückgegeben.

Player

Flash 4 oder höher.

/ (Division)

Syntax

ausdruck1 / *ausdruck2*

Argumente

ausdruck Beliebige Zahl.

Beschreibung

Operator (arithmetisch); dividiert *ausdruck1* durch *ausdruck2*. Die Ausdrucksargumente und die Ergebnisse der Divisionsoperation werden als Gleitkommazahlen mit doppelter Genauigkeit behandelt und ausgegeben.

Player

Flash 4 oder höher.

Beispiel

Diese Anweisung dividiert die Gleitkommazahl 22,0 durch 7,0 und zeigt anschließend das Ergebnis im Ausgabefenster an:

```
trace(22.0 / 7.0);
```

Das Ergebnis ist die Gleitkommazahl 3,1429.

// (Kommentartrennzeichen)

Syntax

`// kommentar`

Argumente

kommentar Text, der kein Bestandteil des Codes ist und vom Interpreter ignoriert werden soll.

Beschreibung

Kommentar; zeigt den Beginn eines Skriptkommentars an. Der gesamte Text, der zwischen dem Kommenartrennzeichen `//` und dem Zeilenendezeichen steht, wird als Kommentar interpretiert und vom ActionScript-Interpreter ignoriert.

Player

Flash 1 oder höher.

Beispiel

In diesem Skript werden Schrägstriche als Kommentartrennzeichen verwendet, um die erste, dritte, fünfte und siebte Zeile als Kommentare auszuweisen:

```
// Festlegen der X-Position der Filmsequenz Ball
ball = getProperty(ball._x);
// Festlegen der Y-Position der Filmsequenz Ball
ball = getProperty(ball._y);
// Festlegen der X-Position der Filmsequenz Kitty
kitty = getProperty(kitty._x);
// Festlegen der Y-Position der Filmsequenz Kitty
kitty_y = getProperty(kitty._y);
```

Siehe auch

„/* (Kommentartrennzeichen)“ auf Seite 201

/* (Kommentartrennzeichen)

Syntax

```
/* kommentar */
/*
* kommentar
* kommentar
*/
```

Argumente

kommentar Beliebiger Text

Beschreibung

Kommentar; zeigt eine oder mehrere Zeilen von Skriptkommentaren an. Der gesamte Text, der zwischen dem öffnenden Kommentarzeichen `/*` und dem schließenden Kommentarzeichen `*/` steht, wird als Kommentar interpretiert und vom ActionScript-Interpreter ignoriert. Verwenden Sie die erstgenannte Syntax zum Kennzeichnen von Kommentaren auf einer einzelnen Zeile und die zweite Syntax zum Kennzeichnen von Kommentaren, die sich über mehrere aufeinanderfolgende Zeilen erstrecken. Wenn Sie bei diesem Typ von Kommentartrennzeichen das schließende Zeichen `*/` auslassen, gibt der ActionScript-Compiler eine Fehlermeldung zurück.

Player

Flash 5 oder höher.

Siehe auch

„`// (Kommentartrennzeichen)`“ auf Seite 201

/= (Divisionszuweisung)

Syntax

`ausdruck1 /= ausdruck2`

Argumente

`ausdruck1,ausdruck2` Ganzzahlen, Gleitkommazahlen oder Zeichenfolgen.

Beschreibung

Operator (Zuweisung); weist *`ausdruck1`* den Wert von *`ausdruck1 / ausdruck2`* zu.

Player

Flash 4 oder höher.

Beispiel

Das folgende Beispiel demonstriert die Verwendung des `/=`-Operators bei Variablen und Zahlen:

```
x /= y ist gleich x = x /y
x = 10;
y = 2;
x /= y;
// x hat nun den Wert 5
```

[] (Arrayzugriffsoperator)

Syntax

`meinArray[„a0", „a1",..."aN“];`

`objekt[wert1, wert2, ...wertN];`

Argumente

meinArray Der Name eines Arrays.

a0, a1,...aN Elemente in einem Array.

wert1, 2,...N Namen von Eigenschaften.

Beschreibung

Operator; erstellt ein neues Objekt und initialisiert die in den Argumenten angegebenen Eigenschaften oder initialisiert ein neues Array mit den Elementen (*a0*), die in den Argumenten angegeben sind.

Der Prototyp des erstellten Objektes ist das generische `Object`-Objekt. Die Verwendung dieses Operators entspricht dem Aufruf von `new Object` und dem anschließendem Zuweisen der Eigenschaften mit Hilfe des Zuweisungsoperators. Dieser Operator kann anstelle des `new`-Operators verwendet werden und bietet Ihnen eine schnelle und einfache Alternative der Objekterstellung.

Player

Flash 4 oder höher.

Beispiel

Die folgenden Codebeispiele stellen zwei unterschiedliche Möglichkeiten zum Erstellen eines neuen und leeren Array-Objektes dar.

```
myArray =[];
myArray = new Array();
```

Das folgende Beispiel stellt einen einfachen Array dar.

```
myArray = ["rot", "orange", "gelb", "grün", "blau", "lila"]
myArray[0]="rot"
myArray[1]="gelb"
myArray[2]="grün"
myArray[3]="blau"
myArray[4]="lila"
```

^ (Bitweises XODER)

Syntax

ausdruck1 ^ *ausdruck2*

Argumente

ausdruck1,ausdruck2 Beliebige Zahl.

Beschreibung

Operator (bitweise); wandelt *ausdruck1* und *ausdruck2* in vorzeichenlose 32-Bit-Ganzzahlen um und gibt an jeder Bit-Position eine 1 zurück, an der die entsprechenden Bits in *ausdruck1* oder *ausdruck2* (jedoch nicht in beiden) 1 sind.

Player

Flash 5 oder höher.

Beispiel

```
15 ^ 9 gibt 6 zurück
(1111 ^ 1001 = 0110)
```

^= (Bitweise XODER-Zuweisung)

Syntax

ausdruck1 `^=` *ausdruck2*

Argumente

ausdruck1,*ausdruck2*Ganzzahlen und Variablen.

Beschreibung

Operator (zusammengesetzte Zuweisung); weist *ausdruck1* den Wert von *ausdruck1* `^` *ausdruck2* zu.

Player

Flash 5 oder höher.

Beispiel

Das folgende Beispiel stellt eine ^=-Operation dar.

```
// 15 dezimal = 1111 binär
x = 15;
// 9 dezimal = 1001 binär
x ^= y;
ergibt
x ^ y (0110 binär)
```

Das folgende Beispiel demonstriert die Verwendung des `^=`-Operators bei Variablen und Zahlen:

```
x ^= y ist gleich x = x ^ y
Wenn x = 15 und y = 9, dann gilt:
15 ^= 9 gibt 6 zurück
```

Siehe auch

„^ (Bitweises XODER)" auf Seite 203

{} (Objektinitialisierung)

Syntax

```
objekt {name1: wert1,
   name1: wert2,
...
   nameN: wertN };
```

Argumente

objekt Das zu erstellende Objekt.

name1,2,...N Der Name der Eigenschaft.

wert1,2,...N Der entsprechende Wert für jede *name*-Eigenschaft.

Beschreibung

Operator; erstellt ein neues Objekt und initialisiert es mit den festgelegten Eigenschaftspaaren aus *name* und *wert*. Der Prototyp des erstellten Objektes ist das generische `Object`-Objekt. Die Verwendung dieses Operators entspricht dem Aufruf von `new Object` und dem anschließendem Zuweisen der Eigenschaftspaaren mit Hilfe des Zuweisungsoperators. Dieser Operator kann anstelle des `new`-Operators verwendet werden und bietet Ihnen eine schnelle und einfache Alternative der Objekterstellung.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Code wird gezeigt, wie ein leeres Objekt mit Hilfe des Objektinitialisierungsoperators und dem Aufruf von `new Object` erstellt werden kann.

```
object = {};
object = new Object();
```

Im Folgenden wird ein Objekt `account` erstellt und die Eigenschaften `name`, `address`, `city`, `state`, `zip` und `balance` initialisiert.

```
account = { name: "Joachim Schmidt",
   address: "Haupstr. 123",
   city: "Bedorf",
   state: "Deutschland",
   zip: "12345",
   balance: "1000" };
```

Im folgenden Beispiel wird die Verschachtelung von Array- und Objektinitialisierung ineinander gezeigt.

```
person = { name: "Peter Piper",
   children: [ "Jakob", "Jenny", "Micha",] };
```

Im folgenden Beispiel wird eine weitere mögliche Verwendung der Daten aus dem obenstehenden Beispiel aufgezeigt, wobei dieselben Ergebnisse erzielt werden.

```
person = new Person();
person.name = 'Joachim Schmidt';
person.children = new Array();
person.children[0] = 'Jakob';
person.children[1] = 'Jenny';
person.children[2] = 'Micha';
```

Siehe auch
„[] (Arrayzugriffsoperator)“ auf Seite 202
„new“ auf Seite 324
„Object (Objekt)“ auf Seite 332

| (Bitweises ODER)

Syntax
ausdruck1|*ausdruck2*

Argumente
ausdruck1,*ausdruck2* Beliebige Zahl.

Beschreibung
Operator (bitweise); wandelt *ausdruck1* und *ausdruck2* in vorzeichenlose 32-Bit-Ganzzahlen um und gibt an jeder Bit-Position eine 1 zurück, an der die entsprechenden Bits entweder in *ausdruck1* oder *ausdruck2* 1 sind.

Player
Flash 5 oder höher.

Beispiel
Das folgende Beispiel stellt eine bitweise ODER-Operation dar. Beachten Sie, dass 15 gleich 1111 binär ist.

```
// 15 dezimal = 1111 binär
x = 15;
// 9 dezimal = 1001 binär
y = 9;
// x | y = binär
z = x | y;
z = 15
```

Im Folgenden wird eine alternative Schreibweise des vorhergehenden Beispiels dargestellt.

```
15 | 9 gibt 15 zurück
(1111 | 0011 = 1111)
```

|| (ODER)

Syntax
ausdruck1 || *ausdruck2*

Argumente
ausdruck1,*ausdruck2* Ein Boolescher Wert oder Ausdruck, der in einen Booleschen Wert umgewandelt werden kann.

Beschreibung

Operator (logisch); wertet *ausdruck1* und *ausdruck2 aus.* . Das Ergebnis ist (`true`), wenn einer oder beide Ausdrücke den Wert `true` haben; das Ergebnis ist nur dann (`false`), wenn beide Ausdrücke den Wert `false` haben.

Bei Ausdrücken, die nicht vom Typ Boolean sind, weist der logische ODER-Operator Flash an, den linken Ausdruck auszuwerten; wenn dieser in `true` umgewandelt werden kann, ist das Ergebnis `true`. Andernfalls wertet er den rechten Ausdruck aus, und das Ergebnis ist der Wert dieses Ausdrucks.

Player

Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird der `||`-Operator in einer `if`-Anweisung verwendet:

```
want = true;
need = true;
love = false;
if (want || need || love){
trace("Zwei von drei ist besser als nichts");
}
```

|= (Bitweise ODER-Zuweisung)

Syntax

ausdruck1 `|=` *ausdruck2*

Argumente

ausdruck1,ausdruck2 Ganzzahlen und Variablen.

Beschreibung

Operator (Zuweisung); weist *ausdruck1* den Wert von *ausdruck1* | *ausdruck2* zu.

Player

Flash 5 oder höher.

Beispiel

Das folgende Beispiel demonstriert die Verwendung des `/=`-Operators bei Variablen und Zahlen:

```
x |= y is gleich x = x | y
Wenn x = 15 und y = 9, dann gilt:
x |= 9 gibt 15 zurück
```

Siehe auch

„| (Bitweises ODER)“ auf Seite 206

~ (Bitweises NICHT)

Syntax
~ *ausdruck*

Argumente
ausdruck Beliebige Zahl.

Beschreibung
Operator (bitweise); wandelt den *ausdruck* in eine vorzeichenlose 32-Bit-Ganzzahl um und invertiert die Bits dann. Das heißt, das Vorzeichen einer Zahl wird geändert und 1 subtrahiert.

Bei einer bitweisen NICHT-Operation wird das Vorzeichen einer Zahl geändert und 1 subtrahiert.

Player
Flash 5 oder höher.

Beispiel
Im Folgenden wird die Durchführung einer bitweisen NICHT-Operation mit einer Variablen numerisch erläutert.

```
~a, ergibt  -1 wenn a = 0, ergibt -2 wenn a = 1, d. h.:
~0=-1 und ~1=-2
```

+ (Addition)

Syntax
ausdruck1 + *ausdruck2*

Argumente
ausdruck1,*ausdruck2* Ganzzahlen, Zahlen, Gleitkommazahlen oder Zeichenfolgen.

Beschreibung
Operator; addiert numerische Ausdrücke oder verkettet Zeichenfolgen. Wenn ein Ausdruck eine Zeichenfolge ist, werden alle Ausdrücke in Zeichenfolgen umgewandelt und verkettet.

Wenn beide Ausdrücke Ganzzahlen sind, ist die Summe eine Ganzzahl; wenn ein oder beide Ausdrücke Gleitkommazahlen sind, ist die Summe eine Gleitkommazahl.

Player
Flash 4; Flash 5 oder höher. In Flash 5 ist + je nach Datentyp des Arguments ein numerischer Operator oder ein Verknüpfungsoperator für Zeichenfolgen. In Flash 4 ist + ausschließlich ein numerischer Operator. Flash 4-Dateien, die in der Flash 5-Erstellungsumgebung verwendet werden, durchlaufen einen Umwandlungsprozess, um die Datentypintegrität zu gewährleisten. Das erste der folgenden Beispiele demonstriert den Umwandlungsprozess.

Beispiel
Im Folgenden wird die Umwandlung einer Flash 4-Datei mit numerischem Qualitätsvergleich demonstriert.

Flash 4-Datei:

```
x + y
```

Umgewandelte Flash 5-Datei:

```
Number(x) + Number(y)
```

Diese Anweisung dividiert die Ganzzahl 2 durch 3 und zeigt anschließend das ganzzahlige Ergebnis 5 im Ausgabefenster an:

```
trace (2 + 3);
```

Diese Anweisung dividiert die Gleitkommazahl 2,5 durch 3,25 und zeigt anschließend als Ergebnis die Gleitkommazahl 5,7500 im Ausgabefenster an:

```
trace (2.5 + 3.25);
```

Diese Anweisung verkettet zwei Zeichenfolgen und zeigt das Ergebnis „Heute ist mein Geburtstag“ im Ausgabefenster an.

```
"Heute ist mein" + "Geburtstag"
```

Siehe auch
„add“ auf Seite 221

+= (Additionszuweisung)

Syntax
ausdruck1 += *ausdruck2*

Argumente
ausdruck1,*ausdruck2* Ganzzahlen, Gleitkommazahlen oder Zeichenfolgen.

Beschreibung
Operator (zusammengesetzte Zuweisung); weist *ausdruck1* dern Wert von *ausdruck1* + *ausdruck2* zu. Dieser Operator führt auch Zeichenfolgenverkettungen durch.

Player
Flash 4 oder höher.

Beispiel
Das folgende Beispiel demonstriert die numerische Verwendung des +=-Operators:

```
x += y ist gleich x = x + y
Wenn x = 5 und y = 10 dann gilt:
x += 10 gibt 15 zurück
```

Das folgende Beispiel demonstriert die Verwendung des +=-Operators bei Zeichenfolgen:

```
x = "Mein Name ist"
x += "Marie"
```

Das Ergebnis des genannten Codes lautet wie folgt:

```
"Mein Name ist Marie"
```

Siehe auch

„+ (Addition)“ auf Seite 208

< (kleiner als)

Syntax

ausdruck1 < ausdruck2

Argumente

ausdruck1,ausdruck2 Zahlen oder Zeichenfolgen.

Beschreibung

Operator (Vergleich); vergleicht zwei Ausdrücke und ermittelt, ob *ausdruck1* kleiner als *ausdruck2* ist (`true`) oder ob *ausdruck1* größer oder gleich *ausdruck2* ist (`false`). Ausdrücke vom Typ Zeichenfolge werden entsprechend der Anzahl an Zeichen in der Zeichenfolge ausgewertet und verglichen.

Player

Flash 4; Flash 5 oder höher. In Flash 5 ist `<` ein Vergleichsoperator, der unterschiedliche Datentypen verarbeiten kann. In Flash 4 ist `<` ein numerischer Operator. Flash 4-Dateien, die in der Flash 5-Erstellungsumgebung verwendet werden, durchlaufen einen Umwandlungsprozess, um die Datentypintegrität zu gewährleisten. Das erste der folgenden Beispiele demonstriert den Umwandlungsprozess.

Beispiel

Im Folgenden wird die Umwandlung einer Flash 4-Datei mit numerischem Qualitätsvergleich demonstriert.

Flash 4-Datei:

```
x < y
```

Umgewandelte Flash 5-Datei:

```
Number(x) < Number(y)
```

Das folgende Beispiel demonstriert die Rückgabewerte `true` und `false` bei Zahlen und Zeichenfolgen:

```
3 < 10 oder "Jo" < "Jakob" gibt true zurück
10 < 3 oder "Jakob" < "Jo" gibt false zurück
```

<< (Bitweises Shift links)

Syntax

ausdruck1 << *ausdruck2*

Argumente

ausdruck1 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, welche/r bitweise nach links verschoben werden soll.

ausdruck2 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, der in eine Ganzzahl zwischen 0 und 31 umgewandelt werden kann.

Beschreibung

Operator (bitweise); wandelt *ausdruck1* und *ausdruck2* in 32-Bit-Ganzzahlen um und verschiebt alle Bits in *ausdruck1* um die Anzahl von Stellen nach links, die durch die Ganzzahl angegeben wird, die sich aus der Umwandlung von *ausdruck2* ergibt. Die Bit-Positionen, die in der Folge dieser Operation leer werden, werden mit 0 aufgefüllt. Das Verschieben eines Werts um eine Stelle nach links entspricht einer Multiplikation mit 2.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird die Ganzzahl `1` um zehn Bit nach links verschoben.

```
x = 1 << 10
```

Das Ergebnis dieser Operation lautet `x = 1024`. Dies berechnet sich wie folgt: 1 dezimal ist gleich 1 binär, 1 binär um 10 nach links verschoben ist 10000000000 binär, und 10000000000 binär ist 1024 dezimal.

Im folgenden Beispiel wird die Ganzzahl `7` um zehn Bit nach links verschoben.

```
x = 7 << 8
```

Das Ergebnis dieser Operation lautet `x = 1792`. Dies berechnet sich wie folgt: 7 dezimal ist gleich 111 binär, 111 binär um 8 nach links verschoben ist 11100000000 binär, und 11100000000 binär ist 1792 dezimal.

Siehe auch

„`>>= (Bitweises Shift rechts und Zuweisung)`“ auf Seite 218

<<= (Bitweises Shift links und Zuweisung)

Syntax

ausdruck1 <<= *ausdruck2*

Argumente

ausdruck1 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, welche/r bitweise nach links verschoben werden soll.

ausdruck2 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, der in eine Ganzzahl zwischen 0 und 31 umgewandelt werden kann.

Beschreibung

Operator (zusammengesetzte Zuweisung); dieser Operator führt eine bitweise Shift-links-Operation durch und speichert den Inhalt als Ergebnis in *ausdruck1*.

Player

Flash 5 oder höher.

Beispiel

Die folgenden beiden Ausdrücke sind äquivalent.

```
A <<= B
A = (A << B)
```

Siehe auch

„<< (Bitweises Shift links)“ auf Seite 211
„>>= (Bitweises Shift rechts und Zuweisung)“ auf Seite 218

<= (kleiner oder gleich)

Syntax

```
ausdruck1 <= ausdruck2
```

Argumente

ausdruck1, *ausdruck2* Zahlen oder Zeichenfolgen.

Beschreibung

Operator (Vergleich); vergleicht zwei Ausdrücke und ermittelt, ob *ausdruck1* kleiner oder gleich *ausdruck2* ist (`true`) oder ob *ausdruck1* größer als *ausdruck2* ist (`false`).

Player

Flash 4; Flash 5 oder höher. In Flash 5 ist `<=` ein Vergleichsoperator, der unterschiedliche Datentypen verarbeiten kann. In Flash 4 ist `<=` ein numerischer Operator. Flash 4-Dateien, die in der Flash 5-Erstellungsumgebung verwendet werden, durchlaufen einen Umwandlungsprozess, um die Datentypintegrität zu gewährleisten. Das erste der folgenden Beispiele demonstriert den Umwandlungsprozess.

Beispiel

Im Folgenden wird die Umwandlung einer Flash 4-Datei mit numerischem Qualitätsvergleich demonstriert.

Flash 4-Datei:

```
x <= y
```

Umgewandelte Flash 5-Datei:

```
Number(x) <= Number(y)
```

Das folgende Beispiel demonstriert die Ergebnisse `true` und `false` bei Zahlen und Zeichenfolgen:

```
5 <= 10 oder "Jo" <= "Jakob" gibt true zurück
10<= 5 oder "Jakob" <= "Jo" gibt false zurück
```

<> (nicht gleich)

Syntax

ausdruck1 `<>` *ausdruck2*

Argumente

ausdruck1, *ausdruck2* Zahlen, Zeichenfolgen, Booleans, Variablen, Objekte, Arrays oder Funktionen.

Beschreibung

Operator (Gleichheit); prüft auf das genaue Gegenteil des ==-Operators. Wenn *ausdruck1* gleich *ausdruck2* ist, ist das Ergebnis `false`. Wie beim ==-Operator hängt die Definition von *gleich* von den verglichenen Datentypen ab.

- Zahlen, Zeichenfolgen und Boolesche Werte werden anhand ihres Werts verglichen.
- Variablen, Objekte, Arrays und Funktionen werden anhand ihrer Referenz verglichen.

Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen `!=`-Operators empfohlen.

Player

Flash 2 oder höher.

Siehe auch

„`!= (nicht gleich)`“ auf Seite 191

= (Zuweisung)

Syntax

ausdruck1 `=` *ausdruck2*

Argumente

ausdruck1 Eine Variable, ein Arrayelement oder eine Objekteigenschaft.

ausdruck2 Ein Wert von einem beliebigen Typ.

Beschreibung

Operator (Zuweisung); weist der Variablen, dem Arrayelement oder der Eigenschaft in *ausdruck1* den Typ von *ausdruck2* (dem rechten Argument) zu.

Player
Flash 4; Flash 5 oder höher. In Flash 5 ist = ein Zuweisungsoperator, und der ==-Operator wird zur Auswertung von Gleichheit verwendet. In Flash 4 ist = ein numerischer Gleichheitsoperator. Flash 4-Dateien, die in der Flash 5-Erstellungsumgebung verwendet werden, durchlaufen einen Umwandlungsprozess, um die Datentypintegrität zu gewährleisten. Das erste der folgenden Beispiele demonstriert den Umwandlungsprozess.

Beispiel
Im Folgenden wird die Umwandlung einer Flash 4-Datei mit numerischem Qualitätsvergleich demonstriert.

Flash 4-Datei:

```
x = y
```

Umgewandelte Flash 5-Datei:

```
Number(x) == Number(y)
```

Im folgenden Beispiel wird der Zuweisungsoperator für die Zuweisung des Datentyps Zahl zur Variable `x` verwendet.

```
x = 5
```

Im folgenden Beispiel wird der Zuweisungsoperator für die Zuweisung des Datentyps Zeichenfolge zur Variable `x` verwendet.

```
x = "Hallo"
```

-= (Negationszuweisung)

Syntax
`ausdruck1 -= ausdruck2`

Argumente
`ausdruck1,ausdruck2` Ganzzahlen, Gleitkommazahlen oder Zeichenfolgen.

Beschreibung
Operator (zusammengesetzte Zuweisung); weist *`ausdruck1`* den Wert von *`ausdruck1 - ausdruck2`* zu.

Player
Flash 4 oder höher.

Beispiel
Das folgende Beispiel demonstriert die Verwendung des `-=`-Operators mit Variablen und Zahlen:

```
x -= y ist gleich x = x - y
Wenn x = 5 und y = 10, dann gilt:
x -= 10 gibt  -5 zurück
```

== (gleich)

Syntax

ausdruck1 == *ausdruck2*

Argumente

ausdruck1,*ausdruck2* Zahlen, Zeichenfolgen, Booleans, Variablen, Objekte, Arrays oder Funktionen.

Beschreibung

Operator (Gleichheit); prüft zwei Ausdrücke auf Gleichheit. Das Ergebnis ist `true`, wenn die Ausdrücke gleich sind.

Die Definition von *Gleichheit* hängt vom Datentyp des Arguments ab:

- Zahlen, Zeichenfolgen und Boolesche Werte werden anhand ihres Werts verglichen und als gleich betrachtet, wenn sie den gleichen Wert haben. Beispielsweise sind zwei Zeichenfolgen gleich, wenn sie die gleiche Anzahl an Zeichen aufweisen.
- Variablen, Objekte, Arrays und Funktionen werden anhand ihrer Referenz verglichen. Zwei Variablen sind gleich, wenn sie auf dasselbe Objekt, denselben Array oder dieselbe Funktion verweisen. Zwei separate Arrays werden nie als gleich betracht, selbst wenn sie dieselbe Anzahl von Elementen aufweisen.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird der ==-Operator in einer `if`-Anweisung verwendet:

```
a = "David" , b = "David";
if (a == b)
trace("David ist David");
```

> (größer als)

Syntax

ausdruck1 > *ausdruck2*

Argumente

ausdruck1,*ausdruck2* Ganzzahlen, Gleitkommazahlen oder Zeichenfolgen.

Beschreibung

Operator (Vergleich); vergleicht zwei Ausdrücke und ermittelt, ob *ausdruck1* größer als *ausdruck2* ist (`true`) oder ob *ausdruck1* kleiner oder gleich *ausdruck2* ist (`false`).

Player
Flash 4; Flash 5 oder höher. In Flash 5 ist > ein Vergleichsoperator, der unterschiedliche Datentypen verarbeiten kann. In Flash 4 ist > ein numerischer Operator. Flash 4-Dateien, die in der Flash 5-Erstellungsumgebung verwendet werden, durchlaufen einen Umwandlungsprozess, um die Datentypintegrität zu gewährleisten. Das folgende Beispiel demonstriert den Umwandlungsprozess.

Beispiel
Im Folgenden wird die Umwandlung einer Flash 4-Datei mit numerischem Qualitätsvergleich demonstriert.

Flash 4-Datei:

```
x > y
```

Umgewandelte Flash 5-Datei:

```
Number(x) > Number(y)
```

>= (größer oder gleich)

Syntax
ausdruck1 >=*ausdruck2*

Argumente
ausdruck1, ausdruck2 Zeichenfolgen, Ganzzahlen oder Gleitkommazahlen.

Beschreibung
Operator (Vergleich); vergleicht zwei Ausdrücke und ermittelt, ob *ausdruck1* größer oder gleich *ausdruck2* ist (`true`) oder ob *ausdruck1* kleiner als *ausdruck2* ist (`false`).

Player
Flash 4; Flash 5 oder höher. In Flash 5 ist >= ein Vergleichsoperator, der unterschiedliche Datentypen verarbeiten kann. In Flash 4 ist >= ein numerischer Operator. Flash 4-Dateien, die in der Flash 5-Erstellungsumgebung verwendet werden, durchlaufen einen Umwandlungsprozess, um die Datentypintegrität zu gewährleisten. Das folgende Beispiel demonstriert den Umwandlungsprozess.

Beispiel
Im Folgenden wird die Umwandlung einer Flash 4-Datei mit numerischem Qualitätsvergleich demonstriert.

Flash 4-Datei:

```
x >= y
```

Umgewandelte Flash 5-Datei:

```
Number(x) >= Number(y)
```

>> (Bitweises Shift rechts)

Syntax

ausdruck1 >> *ausdruck2*

Argumente

ausdruck1 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, welche/r bitweise nach rechts verschoben werden soll.

ausdruck2 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, der in eine Ganzzahl zwischen 0 und 31 umgewandelt werden kann.

Beschreibung

Operator (bitweise); wandelt *ausdruck1* und *ausdruck2* in 32-Bit-Ganzzahlen um und verschiebt alle Bits in *ausdruck1* um die Anzahl von Stellen nach rechts, die durch die Ganzzahl angegeben wird, die sich aus der Umwandlung von *ausdruck2* ergibt. Nach rechts verschobene Bits werden verworfen. Um das Vorzeichen des ursprünglichen Ausdrucks *ausdruck* zu bewahren, werden die Bits auf der linken Seite mit 0 aufgefüllt, wenn das wichtigste Bit (das Bit ganz links) von *ausdruck1* 0 ist. Sie werden mit 1 aufgefüllt, wenn das wichtigste Bit 1 ist. Das Verschieben eines Bits um eine Stelle nach rechts entspricht der Division durch 2 und Verwerfen des Rests.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird 65535 in eine 32-Bit-Ganzzahl umgewandelt und um acht Bit nach rechts verschoben.

```
x = 65535 >> 8
```

Das Ergebnis der genannten Operation lautet wie folgt:

```
x = 255
```

Dies berechnet sich wie folgt: 65535 dezimal ist gleich 1111111111111111 binär (*sechzehnmal 1*), 1111111111111111 binär um acht Bit nach rechts verschoben ist 11111111 binär, und 11111111 binär ist 255 dezimal. Das wichtigste Bit ist 0, da die Ganzzahlen 32-Bit-Zahlen sind, somit ist das Füllbit 0.

Im folgenden Beispiel wird -1 in eine 32-Bit-Ganzzahl umgewandelt und um ein Bit nach rechts verschoben.

```
x = -1 >> 1
```

Das Ergebnis der genannten Operation lautet wie folgt:

```
x = -1
```

Dies berechnet sich wie folgt: -1 dezimal ist gleich 11111111111111111111111111111111 binär (zweiunddreißigmal 1), durch Verschieben um ein Bit nach rechts wird das unwichtigste Bit (das Bit ganz recht) entfernt und das wichtigste Bit mit 1 aufgefüllt. Das Ergebnis ist 11111111111111111111111111111111 (*zweiunddreißigmal 1*) binär, das die 32-Bit-Ganzzahl –1 darstellt.

Siehe auch

„`>>= (Bitweises Shift rechts und Zuweisung)`“ `auf Seite 218`

>>= (Bitweises Shift rechts und Zuweisung)

Syntax

`ausdruck1` `=>>` *`ausdruck2`*

Argumente

`ausdruck1` Eine Zahl, eine Zeichenfolge oder ein Ausdruck, welche/r bitweise nach links verschoben werden soll.

`ausdruck2` Eine Zahl, eine Zeichenfolge oder ein Ausdruck, der in eine Ganzzahl zwischen 0 und 31 umgewandelt werden kann.

Beschreibung

Operator (zusammengesetzte Zuweisung); dieser Operator führt eine bitweise Shift-rechts-Operation durch und speichert den Inhalt als Ergebnis in *`ausdruck1`*.

Player

Flash 5 oder höher.

Beispiel

Die folgenden beiden Ausdrücke sind äquivalent.

```
A >>= B
A = (A >> B)
```

Der folgende kommentierte Code verwendet den bitweisen Operator >>=. Das folgende Beispiel demonstriert zudem alle bitweisen Operatoren.

```
function convertToBinary(number)
{
var result = "";
for (var i=0; i<32; i++) {
// Extrahieren des unwichtigsten Bit mit Hilfe von bitweisem UND
var lsb = number & 1;
// Hinzufügen dieses Bit zur Ergebniszeichenfolge
result = (lsb ? "1" : "0") + result;
// Verschieben der Zahl um ein Bit nach rechts, um das nächste Bit
anzuzeigen
}number >>= 1;
return result;
}
convertToBinary(479)
//Zurückgeben der Zeichenfolge
00000000000000000000000111011111
//Die obenstehende Zeichenfolge ist die binäre Darstellung der
Dezimalzahl 479.
```

Siehe auch

„<< (Bitweises Shift links)“ auf Seite 211

>>> (Vorzeichenloses bitweises Shift rechts)

Syntax

ausdruck1 >>> *ausdruck2*

Argumente

ausdruck1 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, welche/r bitweise nach rechts verschoben werden soll.

ausdruck2 Eine Zahl, eine Zeichenfolge oder ein Ausdruck, der in eine Ganzzahl zwischen 0 und 31 umgewandelt werden kann.

Beschreibung

Operator (bitweise); entspricht dem Operator für bitweises Shift rechts (>>), bewahrt jedoch nicht das Vorzeichen des ursprünglichen Ausdrucks *ausdruck*, da die Bits auf der linken Seite stets mit 0 aufgefüllt werden.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird -1 in eine 32-Bit-Ganzzahl umgewandelt und um ein Bit nach rechts verschoben.

```
x = -1 >>> 1
```

Das Ergebnis der genannten Operation lautet wie folgt:

```
x = 2147483647
```

Dies berechnet sich wie folgt: -1 dezimal ist gleich 11111111111111111111111111111111 binär (*zweiunddreißigmal 1*), beim vorzeichenlosen Verschieben um ein Bit nach rechts wird das unwichtigste (äußerste rechte) Bit entfernt und das wichtigste (äußerste linke) Bit mit einer 0 aufgefüllt. Das Ergebnis lautet wie folgt:

01111111111111111111111111111111 binär,

das für die 32-Bit-Ganzzahl `2147483647` steht.

Siehe auch

„>>= (Bitweises Shift rechts und Zuweisung)" auf Seite 218

>>>= (Vorzeichenloses bitweises Shift rechts und Zuweisung)

Syntax

`ausdruck1` `>>>=` *`ausdruck2`*

Argumente

`ausdruck1` Eine Zahl, eine Zeichenfolge oder ein Ausdruck, welche/r bitweise nach links verschoben werden soll.

`ausdruck2` Eine Zahl, eine Zeichenfolge oder ein Ausdruck, der in eine Ganzzahl zwischen 0 und 31 umgewandelt werden kann.

Beschreibung

Operator (zusammengesetzte Zuweisung); führt eine vorzeichenlose bitweise Shift-rechts-Operation durch und speichert den Inhalt als Ergebnis in *`ausdruck1`*.

Player

Flash 5 oder höher.

Beispiel

Die folgenden beiden Ausdrücke sind äquivalent.

```
A >>>= B
A = (A >>> B)
```

Siehe auch

„>>> (Vorzeichenloses bitweises Shift rechts)" auf Seite 219
„>>= (Bitweises Shift rechts und Zuweisung)" auf Seite 218

add

Syntax

`zeichenfolge1 add zeichenfolge2`

Argumente

zeichenfolge1,2 Beliebige Zeichenfolge.

Beschreibung

Operator; verkettet zwei oder mehr Zeichenfolgen. Der `add`-Operator ersetzt den `&`-Operator aus Flash 4; Flash 4-Dateien, die den `&`-Operator verwenden, werden bei Verwendung in der Flash 5-Erstellungsumgebung automatisch so umgewandelt, dass sie den `add`-Operator für die Zeichenfolgenverkettung verwenden. Der `add`-Operator ist jedoch in Flash 5 veraltet, und für die Erstellung von Inhalten für den Flash 5 Player wird die Verwendung des +-Operators empfohlen. Verwenden Sie den `add`-Operator zum Verketten von Zeichenfolgen, wenn Sie Inhalte für den Flash 4 Player oder frühere Player-Versionen erstellen.

Player

Flash 4 oder höher.

Siehe auch

„+ (Addition)“ auf Seite 208

_alpha

Syntax

```
instanzname._alpha
instanzname._alpha = wert;
```

Argumente

instanzname Der Name der Instanz einer Filmsequenz.

wert Eine Zahl von 0 bis 100 zur Angabe der Alpha-Transparenz.

Beschreibung

Eigenschaft; setzt oder ruft die Alpha-Transparenz (*wert*) der Filmsequenz ab. Gültige Werte sind 0 (vollständig transparent) bis 100 (undurchsichtig). Die Objekte einer Filmsequenz mit `_alpha` von 0 sind aktiv, auch wenn sie nicht sichtbar sind. Beispielsweise können Sie auf eine Schaltfläche in einer Filmsequenz klicken, obwohl deren `_alpha`-Eigenschaft auf 0 gesetzt ist.

Player

Flash 4 oder höher.

Beispiel

Die folgenden Anweisungen setzen die `_alpha`-Eigenschaft einer Filmsequenz mit dem Namen `star` auf 30 %, wenn auf die Schaltfläche geklickt wird.

```
on(release) {
   setProperty(star._alpha = 30);
}
```

oder

```
on(release) {
   star._alpha = 30;
}
```

and

Syntax

bedingung1 `and` *bedingung2*

Argumente

bedingung1,bedingung2 Bedingungen oder Ausdrücke, die den Wert `true` oder `false` haben können.

Beschreibung

Operator; führt eine logische UND-Operation im Flash 4 Player aus. Wenn beide Ausdrücke `true` ergeben, ist der gesamte Ausdruck `true`.

Player

Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen `&&`-Operators empfohlen.

Siehe auch

„&& (kurzgeschlossenes UND)“ auf Seite 194

Array (Objekt)

Das Array-Objekt ermöglicht den Zugriff auf und die Bearbeitung von Arrays. Ein Array ist ein Objekt, dessen Eigenschaften durch eine Zahl angegeben werden, die für die entsprechende Position im Array steht. Diese Zahl wird auch als Index bezeichnet. Alle Arrays haben die Basis Null, d. h., das erste Element im Array ist [0], das zweite Element [1] usw. Im folgenden Beispiel enthält `myArray` die Monate des Jahres, die durch Zahlen angegeben werden.

```
myArray[0] = "Januar"
myArray[1] = "Februar"
myArray[2] = "März"
myArray[3] = "April"
```

Zum Erstellen eines neuen Array-Objektes verwenden Sie den Konstruktor `new Array`. Für den Zugriff auf die Elemente eines Arrays verwenden Sie den Arrayzugriffsoperator `[ ]`.

Zusammenfassung der Methoden für das Array-Objekt

Methode	Beschreibung
`concat`	Verkettet die Argumente und gibt sie als neues Array zurück.
`join`	Verbindet alle Elemente eines Arrays zu einer Zeichenfolge.
`pop`	Entfernt das letzte Element eines Arrays und gibt seinen Wert zurück.
`push`	Fügt am Ende eines Arrays ein oder mehrere Elemente hinzu und gibt die neue Länge des Arrays zurück.
`reverse`	Kehrt die Richtung eines Arrays um.
`shift`	Entfernt das erste Element eines Arrays und gibt seinen Wert zurück.
`slice`	Extrahiert einen Abschnitt eines Arrays und gibt ihn als neues Array zurück.
`sort`	Sortiert ein Array in sich selbst.
`splice`	Fügt einem Array Elemente hinzu und/oder entfernt diese.
`toString`	Gibt einen Zeichenfolgenwert mit den Elementen im Array-Objekt zurück.
`unshift`	Fügt am Anfang eines Arrays ein oder mehrere Elemente hinzu und gibt die neue Länge des Arrays zurück.

Zusammenfassung der Eigenschaften für das Array-Objekt

Eigenschaft	Beschreibung
`length`	Gibt die Länge des Arrays zurück.

Konstruktor für das Array-Objekt

Syntax

```
new Array();

new Array(länge);

new Array(element0, element1, element2,...elementN);
```

Argumente

länge Eine Ganzzahl, die die Anzahl der Elemente im Array angibt. Bei nicht fortlaufenden Elementen gibt die Länge die Indexnummer des letzten Elements im Array plus 1 an. Weitere Informationen erhalten Sie unter der Beschreibung der `Array.length`-Eigenschaft.

element0...elementN Eine Liste von mindestens zwei beliebigen Werten. Die Werte können Zahlen, Namen oder sonstige in einem Array festgelegte Elemente sein. Das erste Element in einem Array hat den Index oder die Position 0.

Beschreibung

Konstruktor; ermöglicht den Zugriff auf und die Bearbeitung von Elementen in einem Array. Arrays haben die Basis Null, und die Elemente werden nach ihrer Ordnungszahl indiziert.

Wenn Sie keine Argumente angeben, wird ein Array mit der Länge 0 erstellt.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird ein neues Array-Objekt mit einer Ausgangslänge von 0 erstellt.

```
myArray = new Array();
```

Im folgenden Beispiel wird das neue Array-Objekt `A-Team` mit einer Ausgangslänge von 4 erstellt.

```
A-Team = new Array("Jody", "Mary", "Marcelle", "Judy");
```

Die Ausgangselemente des Arrays `A-Team` lauten wie folgt:

```
myArray[0] = "Jody"
myArray[1] = "Mary"
myArray[2] = "Marcelle"
myArray[3] = "Judy"
```

Siehe auch

„`Array.length`“ auf Seite 226

Array.concat

Syntax

meinArray`.concat(`*wert0,wert1,...wertN*`);`

Argumente

wert0,...wertN Zahlen, Elemente oder Zeichenfolgen, die in einem neuen Array verkettet werden sollen.

Beschreibung

Methode; verkettet die in den Argumenten angegebenen Elemente, soweit vorhanden, erstellt einen neuen Array und gibt diesen zurück. Wenn die Argumente ein Array angeben, werden anstelle des Arrays selbst die Elemente dieses Arrays verkettet.

Player

Flash 5 oder höher.

Beispiel

Mit dem folgenden Code werden zwei Arrays verkettet:

```
alpha = new Array("a","b","c");
numeric = new Array(1,2,3);
alphaNumeric=alpha.concat(numeric); // Erstellt Array
["a","b","c",1,2,3]
```

Mit dem folgenden Code werden drei Arrays verkettet:

```
num1=[1,3,5];
num2=[2,4,6];
num3=[7,8,9];
nums=num1.concat(num2,num3) // erstellt Array [1,3,5,2,4,6,7,8,9]
```

Array.join

Syntax

```
meinArray.join();
meinArray.join(separator);
```

Argumente

separator Ein Zeichen oder eine Zeichenfolge, mit der die Arrayelemente in der zurückgegebenen Zeichenfolge voneinander getrennt werden. Wenn Sie dieses Argument auslassen, wird als Standardseparator das Komma verwendet.

Beschreibung

Methode; wandelt die Elemente in einem Array in Zeichenfolgen um, verkettet sie, fügt zwischen den Elementen den angegebenen Separator ein und gibt die resultierende Zeichenfolge zurück.

Player

Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird ein Array mit drei Elementen erstellt. Das Array wird anschließend dreimal verbunden: mit dem Standardseparator, anschließend mit einem Komma und Leerzeichen und dann mit einem Pluszeichen.

```
a = new Array("Erde","Mond","Sonne")
// weist myVar1 "Erde1,Mond,Sonne" zu
myVar1=a.join();
// weist myVar2 "Erde, Mond, Sonne" zu
myVar2=a.join(", ");
// weist myVar3 "Erde + Mond + Sonne" zu
myVar3=a.join(" + ");
```

Array.length

Syntax
meinArray.length;

Argumente
Keine.

Beschreibung
Eigenschaft; enthält die Länge des Arrays. Diese Eigenschaft wird automatisch aktualisiert, wenn dem Array neue Elemente hinzugefügt werden. Bei der Zuweisung `meinArray[index] = wert`, wobei `index` eine Zahl und `index+1` größer als die `length`-Eigenschaft ist, wird die `length`-Eigenschaft auf `index + 1` aktualisiert.

Player
Flash 5 oder höher.

Beispiel
Mit dem folgenden Code wird die Aktualisierung der length-Eigenschaft erläutert.

```
//Ausgangslänge ist 0
myArray = new Array();
//myArray.length wird auf 1 aktualisiert
myArray[0] = 'a'
//myArray.length wird auf 2 aktualisiert
myArray[1] = 'b'
//myArray.length wird auf 10 aktualisiert
myArray[9] = 'c'
```

Array.pop

Syntax
meinArray.pop();

Argumente
Keine.

Beschreibung
Methode; entfernt das letzte Element aus einem Array und gibt den Wert dieses Elements zurück.

Player
Flash 5 oder höher.

Beispiel
Mit dem folgenden Code wird das Array `myPets` mit vier Elementen erstellt und anschließend das letzte Element entfernt.

```
myPets = ["Katze", "Hund", "Vogel", "Fisch"];
popped = myPets.pop();
```

Array.push

Syntax
meinArray.push(*wert,...*);

Argumente
wert Ein oder mehrere Werte, die an das Array angehängt werden.

Beschreibung
Methode; fügt am Ende eines Arrays ein oder mehrere Elemente hinzu und gibt die neue Länge des Arrays zurück.

Player
Flash 5 oder höher.

Beispiel
Mit dem folgenden Code wird das Array `myPets` mit zwei Elementen erstellt, dem anschließend zwei Elemente hinzugefügt werden. Nach der Ausführung des Codes hat `pushed` den Wert 4.

```
myPets = ["Katze", "Hund"];
pushed = myPets.push("Vogel", "Fisch")
```

Array.reverse

Syntax
meinArray.reverse();

Argumente
Keine.

Beschreibung
Methode; kehrt die Reihenfolge der Arrayelemente im Array selbst um.

Player
Flash 5 oder höher.

Beispiel
Das folgende Beispiel demonstriert die Verwendung der `Array.reverse`-Methode.

```
var numbers = [1, 2, 3, 4, 5, 6];
trace(numbers.join())
 numbers.reverse()
 trace(numbers.join())
```

Ausgabe:

```
 1,2,3,4,5,6
 6,5,4,3,2,1
```

Array.shift

Syntax
meinArray.shift();

Argumente
Keine.

Beschreibung
Methode; entfernt das erste Element aus einem Array und gibt dieses Element zurück.

Player
Flash 5 oder höher.

Beispiel
Mit dem folgenden Code wird das Array `myPets` erstellt, aus dem anschließend das erste Element entfernt wird:

```
myPets = ["Katze", "Hund", "Vogel", "Fisch"];
shifted = myPets.shift();
```

Der Rückgabewert lautet `Katze`.

Siehe auch
„Array.pop“ auf Seite 227

„Array.unshift“ auf Seite 232

Array.slice

Syntax

```
meinArray.slice(start, ende);
```

Argumente

start Eine Zahl, die den Index des Segmentanfangs angibt. Wenn *start* eine negative Zahl ist, liegt der Ausgangspunkt am Ende des Arrays, wobei -1 das letzte Element ist.

ende Eine Zahl, die den Index des Segmentendes angibt. Wenn Sie dieses Argument auslassen, umfasst das Segment alle Elemente vom Anfang bis zum Ende des Arrays. Wenn *ende* eine negative Zahl ist, wird der Endpunkt vom Ende des Arrays aus berechnet, wobei -1 das letzte Element ist.

Beschreibung

Methode; extrahiert ein Segment oder eine Teilzeichenfolge des Arrays und gibt dies als neues Array zurück, ohne das Ausgangsarray zu verändern. Das zurückgegebene Array umfasst das Element *start* und alle Elemente bis zum Element *ende* (ausschließlich).

Player

Flash 5 oder höher.

Array.sort

Syntax

```
meinArray.sort();
meinArray.sort(sortfunktion);
```

Argumente

sortfunktion Eine optionale Vergleichsfunktion, mit der die Sortierreihenfolge festgelegt wird. Mit den angenommenen Argumenten A und B führt die angegebene Sortierfunktion folgenden Sortiervorgang durch:

- -1, wenn A in der Sortierreihenfolge vor B liegt
- 0, wenn A = B
- 1, wenn A in der Sortierreihenfolge hinter B liegt

Beschreibung

Methode; sortiert das Array in sich selbst, ohne eine Kopie anzulegen. Wenn Sie das Argument *sortfunktion* auslassen, sortiert Flash die Elemente in sich selbst unter Verwendung des <-Vergleichsoperators.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird `Array.sort` ohne Angabe des Arguments *`sortfunktion`* verwendet.

```
var fruits = ["Orangen", "Ananas", "Stachelbeeren",
                "Pfirsiche", "Kirschen"];
 trace(fruits.join())
 fruits.sort()
 trace(fruits.join())
```

Ausgabe:

```
 Orangen,Ananas,Stachelbeeren,Pfirsiche,Kirschen
 Ananas,Kirschen,Orangen,Pfirsiche,Stachelbeeren
```

Im folgenden Beispiel wird `array.sort` mit einer angegebenen Sortierfunktion verwendet.

```
var passwords = [
   "gary:foo",
   "mike:bar",
   "john:snafu",
   "steve:yuck",
   "daniel:1234"
 ];
 function order (a, b) {
   // Zu sortierende Einträge in der Form
   // Name:Kennwort
   // Sortierung mit dem Namensteil des
   // Eintrags als einzigem Sortierschlüssel.
   var name1 = a.split(':')[0];
   var name2 = b.split(':')[0];
   if (name1 < name2) {
     return -1;
   } else if (name1 > name2) {
     return 1;
   } else {
     return 0;
   }
 }
 for (var i=0; i< password.length; i++) {
   trace (passwords.join());
 }
 passwords.sort(order);
 trace ("Sortiert:")
 for (var i=0; i< password.length; i++) {
   trace (passwords.join());
 }
```

Ausgabe:

```
 daniel:1234
 gary:foo
 john:snafu
 mike:bar
 steve:yuck
```

Array.splice

Syntax

meinArray.splice(*start*, *löschanzahl*, *wert0*,*wert1*...*wertN*);

Argumente

start Der Index des Arrayelements, mit dem der Einfüge- und/oder Löschvorgang beginnt.

löschanzahl Die Anzahl der zu löschenden Elemente. Diese Zahl umfasst das im Argument *start* angegebene Element. Wenn für *löschanzahl* kein Wert angegeben wurde, löscht die Methode alle Werte vom Element *start* bis zum letzten Arrayelement.

wert Kein, ein oder mehrere Werte, die an der im Argument *start* angegebenen Position in das Array eingefügt werden. Dieses Argument ist optional.

Beschreibung

Methode; fügt einem Array Elemente hinzu und/oder entfernt diese. Diese Methode nimmt Änderungen am Array selbst vor, ohne eine Kopie anzulegen.

Player

Flash 5 oder höher.

Array.toString

Syntax

meinArray.toString();

Argumente

Keine.

Beschreibung

Methode; gibt einen Zeichenfolgenwert mit den Elementen im angegebenen Array-Objekt zurück. Jedes Element des Arrays von Index 0 bis Index *meinArray*.length-1 wird in eine verkettete, durch Komma getrennte Zeichenfolge umgewandelt.

Player

Flash 5 oder höher.

Beispiel

Das folgende Beispiel erstellt myArray und wandelt es in eine Zeichenfolge um.

```
myArray = new Array();
myArray[0] = 1;
myArray[1] = 2;
myArray[2] = 3;
myArray[3] = 4;
myArray[4] = 5;

trace(myArray.toString())
```

Ausgabe:

```
1,2,3,4,5
```

Array.unshift

Syntax

meinArray.unshift(*wert1*,*wert2*,...*wertN*);

Argumente

wert1,...wertN Eine oder mehrere Zahlen, Elemente oder Variablen, die am Anfang des Arrays eingefügt werden sollen.

Beschreibung

Methode; fügt am Anfang eines Arrays ein oder mehrere Elemente hinzu und gibt die neue Länge des Arrays zurück.

Player

Flash 5 oder höher.

Boolean (Funktion)

Syntax

Boolean(*ausdruck*);

Argumente

ausdruck Die Variable, Zahl oder Zeichenfolge, die in einen Boolean umgewandelt werden soll.

Beschreibung

Funktion; wandelt die angegebenen Argumente in den Typ Boolean um und gibt den Booleschen Wert zurück.

Player

Flash 5 oder höher.

Boolean (Objekt)

Das Boolean-Objekt ist ein einfaches Wrapperobekt mit den gleichen Funktionen wie für das Boolean-Standardobjekt in JavaScript. Mit dem Boolean-Objekt können Sie den Grunddatentyp oder die Darstellung als Zeichenfolge des Boolean-Objektes abrufen.

Zusammenfassung der Methoden für das Boolean-Objekt

Methode	Beschreibung
`toString`	Gibt die Darstellung als Zeichenfolge `(true)` oder `(false)` des Boolean-Objektes zurück.
valueOf	Gibt den Grundwerttyp des angegebenen Boolean-Objektes zurück.

Konstruktor für das Boolean-Objekt

Syntax

```
new Boolean();
```

```
new Boolean(x);
```

Argumente

x Zahl, Zeichenfolge, Boolean, Objekt, Filmsequenz oder anderer Ausdruck. Dieses Argument ist optional.

Beschreibung

Konstruktor; erstellt eine Instanz des Boolean-Objektes. Wenn Sie das Argument *x* auslassen, wird das Boolean-Objekt mit dem Wert `false` initialisiert. Wenn Sie *x* angeben, wertet die Methode das Argument aus und gibt das Ergebnis als Booleschen Wert zurück. Dabei werden die folgenden Zuordnungsregeln angewendet.

- Bei einer Zahl *x* gibt die Funktion `true` zurück, wenn *x* nicht gleich 0 ist; wenn *x* eine andere Zahl ist, gibt die Funktion `false` zurück.
- Wenn *x* ein Boolean ist, gibt die Funktion *x* zurück.
- Bei einem Objekt oder einer Filmsequenz *x* gibt die Funktion `true` zurück, wenn *x* nicht gleich `null` ist; andernfalls gibt die Funktion `false` zurück.
- Bei einer Zeichenfolge *x* gibt die Funktion `true` zurück, wenn `Number(x)` nicht gleich 0 ist; andernfalls gibt die Funktion `false` zurück.

Anmerkung: Um die Kompabilität mit Flash 4 zu gewährleisten, entspricht die Verarbeitung von Zeichenfolgen durch das Boolean-Objekt nicht dem ECMA-262-Standard.

Player

Flash 5 oder höher.

Boolean.toString

Syntax
`Boolean.toString();`

Argumente
Keine.

Beschreibung
Methode; gibt die Darstellung als Zeichenfolge `true` oder `false` des Boolean-Objektes zurück.

Player
Flash 5 oder höher.

Boolean.valueOf

Syntax
`Boolean.valueOf();`

Argumente
Keine.

Beschreibung
Methode; gibt den Grundwerttyp des angegebenen Boolean-Objektes zurück und wandelt das Boolean-Wrapperobjekt in diesen Grundwerttyp um.

Player
Flash 5 oder höher.

break

Syntax
`break;`

Argumente

Keine.

Beschreibung
Aktion; wird in Schleifen (`for`, `for..in`, `do...while` oder `while`) verwendet. Die `break`-Aktion weist Flash an, den restlichen Teil der Schleife zu überspringen, die Schleifenaktion abzubrechen und die Anweisung auszuführen, die auf die Schleifenanweisung folgt. Mit der `break`-Aktion können Sie eine Reihe von verschachtelten Schleifen verlassen.

Player
Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird die `break`-Aktion verwendet, um eine Endlosschleife zu verlassen.

```
i = 0;
while (true) {
  if (i >= 100) {
    break;
  }
  i++;
}
```

call

Syntax

`call(`*`bild`*`);`

Argumente

`bild` Der Name oder die Nummer des Bilds, das als Kontext des Skripts aufgerufen werden soll.

Beschreibung

Aktion; übergibt den Kontext des aktuellen Skripts an das mit dem aufgerufenen Bild verbundene Skript. Lokale Variablen existieren nach Beendigung des Skripts nicht mehr.

Player

Flash 4 oder höher. Diese Aktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `function`-Aktion empfohlen.

Siehe auch

„function“ auf Seite 269

chr

Syntax

`chr(`*`nummer`*`);`

Argumente

`nummer` Die ASCII-Codenummer, die in ein Zeichen umgewandelt werden soll.

Beschreibung

Zeichenfolgenfunktion; wandelt ASCII-Codenummern in Zeichen um.

Player

Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `String.fromCharCode`-Methode empfohlen.

Beispiel
Im folgenden Beispiel wird die Zahl 65 in den Buchstaben „A“ umgewandelt.

```
chr(65) = "A"
```

Siehe auch
„String.fromCharCode“ auf Seite 369

Color (Objekt)

Mit dem Color-Objekt können Sie den RGB-Farbwert und die Farbtransformation einer Filmsequenz setzen und abrufen. Das Color-Objekt wird von Flash Player ab Version 5 unterstützt.

Vor einem Aufruf der Methoden des Color-Objektes muss der Konstruktor `new Color()` zum Erstellen einer Instanz des Color-Objektes verwendet werden.

Zusammenfassung der Methoden für das Color-Objekt

Methode	Beschreibung
`getRGB`	Gibt den numerischen RGB-Wert zurück, der durch den letzten Aufruf von `setRGB` gesetzt wurde.
getTransform	Gibt die Transformationsinformationen zurück, die durch den letzten Aufruf von `setTransform` gesetzt wurden.
`setRGB`	Setzt die hexadezimale Darstellung des RGB-Werts für ein Color-Objekt.
`setTransform`	Setzt die Farbtransformation für ein Color-Objekt.

Konstruktor für das Color-Objekt

Syntax

```
new Color(ziel);
```

Argumente
`ziel` Der Name der Filmsequenz, der die neue Farbe zugewiesen werden soll.

Beschreibung
Konstruktor; erstellt ein Color-Objekt für die im Argument *`ziel`* angegebene Filmsequenz.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird ein neues Color-Objekt mit dem Namen `myColor` für den Film `myMovie` erstellt.

```
myColor = new Color(myMovie);
```

Color.getRGB

Syntax
`myColor.getRGB();`

Argumente
Keine.

Beschreibung
Methode; gibt die numerischen Werte zurück, die durch den letzten Aufruf von `setRGB` gesetzt wurden.

Player
Flash 5 oder höher.

Beispiel
Der folgende Code ruft den RGB-Wert als hexadezimale Zeichenfolge ab:

```
value = (getRGB()).toString(16);
```

Siehe auch
„Color.setRGB“ auf Seite 237

Color.getTransform

Syntax
`myColor.getTransform();`

Argumente
Keine.

Beschreibung
Methode; gibt den Transformationswert zurück, der durch den letzten Aufruf von `setTransform` gesetzt wurde.

Player
Flash 5 oder höher.

Siehe auch
„Color.setTransform“ auf Seite 238

Color.setRGB

Syntax
`myColor.setRGB(0xRRGGBB);`

Argumente
0xRRGGBB Die hexadezimale oder RGB-Farbe, die gesetzt werden soll. *RR*, *GG* und *BB* setzen sich jeweils aus zwei Hexadezimalziffern zusammen, die den Offset für jede Farbkomponente festlegen.

Beschreibung
Methode; gibt eine RGB-Farbe für das Color-Objekt an. Ein Aufruf dieser Methode überschreibt alle vorherigen Einstellungen durch die `setTransform`-Methode.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird der RGB-Farbwert für die Filmsequenz `myMovie` gesetzt.

```
myColor = new Color(myMovie);
myColor.setRGB(0x993366);
```

Siehe auch
„`Color.setTransform`“ auf Seite 238

Color.setTransform

Syntax
`myColor``.setTransform(`*`colorTransformObjekt`*`);`

Argumente
`colorTransformObjekt` Ein Objekt, das mit dem Konstruktor des generischen Object-Objektes erstellt wird und die Farbtransformationswerte als Parameter angibt. Das Color Transform-Objekt muss über die Parameter `ra`, `rb`, `ga`, `gb`, `ba`, `bb`, `aa`, `ab` verfügen, die im Folgenden erläutert werden.

Beschreibung
Methode; setzt die Farbtransformationsinformationen für ein Color-Objekt. Das Argument *`colorTransformObjekt`* ist ein Objekt, das mit Hilfe des generischen Object-Objektes mit Parametern erstellt wird, die Prozent- und Offset-Werte für die Rot-, Grün-, Blau- und Alpha-(Transparenz-)komponenten einer Farbe angeben. Diese werden in einem Format *0xRRGGBBAA* eingegeben.

Die Parameter für ein Color Transform-Objekt sind wie folgt definiert:

- *`ra`* ist der Prozentsatz für die Rotkomponente (-100 bis 100).
- *`rb`* ist der Offset für die Rotkomponente (-255 bis 255).
- *`ga`* ist der Prozentsatz für die Grünkomponente (-100 bis 100).
- *`gb`* ist der Offset für die Grünkomponente (-255 bis 255).
- *`ba`* ist der Prozentsatz für die Blaukomponente (-100 bis 100).
- *`bb`* ist der Offset für die Blaukomponente (-255 bis 255).
- *`aa`* ist der Prozentsatz für den Alpha (-100 bis 100).
- *`ab`* ist der Offset für den Alpha (-255 bis 255).

Ein Color Transform-Objekt wird wie folgt erstellt:

```
myColorTransform = new Object();
myColorTransform.ra = 50;
myColorTransform.rb = 244;
myColorTransform.ga = 40;
myColorTransform.gb = 112;
myColorTransform.ba = 12;
myColorTransform.bb = 90;
myColorTransform.aa = 40;
myColorTransform.ab = 70;
```

Sie können auch die folgende Syntax verwenden:

```
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba:
'12', bb: '90', aa: '40', ab: '70'}
```

Player

Flash 5 oder höher.

Beispiel

Das folgende Beispiel demonstriert das Verfahren zum Erstellen eines neuen Color-Objektes für einen Zielfilm, dem Erstellen eines Color Transform-Objektes mit den Parametern, die oben mit dem Object-Konstruktor definiert wurden, und das Übergeben des Color Transform-Objektes an ein Color-Objekt mit der `setTransform`-Methode.

```
//Erstellen eines Color-Objektes namens myColor für das Ziel
myMovie
myColor = new Color(myMovie);
//Erstellen eines Color Transform-Objektes namens myColorTransfrom
mit Hilfe des generischen Object-Objekts
myColorTransform = new Object;
// Setzen der Werte für myColorTransform
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba:
'12', bb: '90', aa: '40', ab: '70'}
//Zuordnen des Color Transform-Objektes zu dem für myMovie
erstellten Color-Objekt
myColor.setTransform(myColorTransform);
```

continue

Syntax

`continue;`

Argumente

Keine.

Beschreibung

Aktion; wird in unterschiedlichen Typen von Schleifenanweisungen verwendet.

In einer `while`-Schleife weist `continue` Flash an, den restlichen Teil der Schleife zu überspringen, an den Anfang der Schleife zu springen und die Bedingung zu prüfen.

In einer `do...while` -Schleife weist `continue` Flash an, den restlichen Teil der Schleife zu überspringen, an das Ende der Schleife zu springen und die Bedingung zu prüfen.

In einer `for` -Schleife weist `continue` Flash an, den restlichen Teil der Schleife zu überspringen und zur Auswertung des Bis-Ausdrucks der `for`-Schleife zu springen.

In einer `for...in` -Schleife weist `continue` Flash an, den restlichen Teil der Schleife zu überspringen, zurück an den Anfang der Schleife zu springen und den nächsten Wert in der Aufzählung zu verarbeiten.

Player
Flash 4 oder höher.

Siehe auch
„do... while“ auf Seite 260
„for“ auf Seite 266
„for..in“ auf Seite 267
„while“ auf Seite 383

_currentframe

Syntax
instanzname`._currentframe`

Argumente
instanzname Der Name der Instanz einer Filmsequenz.

Beschreibung
Eigenschaft (schreibgeschützt); gibt die Nummer des Bilds zurück, bei dem sich der Abspielkopf in der Zeitleiste momentan befindet.

Player
Flash 4 oder höher.

Beispiel
Im folgenden Beispiel wird eine Filmsequenz mit `_currentframe` angewiesen, um fünf Bilder weiter als das Bild zu springen, das die Aktion enthält:

```
gotoAndStop(_currentframe + 5);
```

Date (Objekt)

Mit dem Date-Objekt können Sie Daten- und Zeitwerte relativ zur Weltzeit (Mittlere Greenwich-Zeit (GMT), heute als Koordinierte Weltzeit (UTC) bezeichnet) oder relativ zur Zeit des Betriebssystems abrufen, auf dem der Flash Player ausgeführt wird. Um die Methoden des Date-Objektes aufzurufen, müssen Sie zunächst mit dem Konstruktor eine Instanz des Date-Objektes erstellen.

Das Date-Objekt benötigt den Flash 5 Player.

Die Methoden des Date-Objektes sind nicht statisch, sondern gelten nur für die einzelne Instanz des Date-Objektes, die beim Methodenaufruf angegeben wird.

Zusammenfassung der Methoden für das Date-Objekt

Methode	Beschreibung
`getDate`	Gibt den Tag des Monats des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getDay`	Gibt den Tag des Monats des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getFullYear`	Gibt das vierstellige Jahr des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getHours`	Gibt die Stunde des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getMilliseconds`	Gibt die Millisekunden des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getMinutes`	Gibt die Minuten des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getMonth`	Gibt den Monat des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getSeconds`	Gibt die Sekunden des angegebenen Date-Objektes nach lokaler Zeit zurück.
`getTime`	Gibt für das angegebene Date-Objekt die Anzahl der Millisekunden zurück, die seit Mitternacht des 1. Januars 1970 Weltzeit vergangen sind.
`getTimezoneOffset`	Gibt die Differenz zwischen der lokalen Zeit des Computers und der Weltzeit in Minuten zurück.
`getUTCDate`	Gibt den Tag (das Datum) des Monats des angegebenen Date-Objektes nach Weltzeit zurück.
`getUTCDay`	Gibt den Wochentag des angegebenen Date-Objektes nach Weltzeit zurück.
`getUTCFullYear`	Gibt das vierstellige Jahr des angegebenen Date-Objektes nach Weltzeit zurück.
`getUTCHours`	Gibt die Stunde des angegebenen Date-Objektes nach Weltzeit zurück.
`getUTCMilliseconds`	Gibt die Millisekunden des angegebenen Date-Objektes nach Weltzeit zurück.
`getUTCMinutes`	Gibt die Minute des angegebenen Date-Objektes nach Weltzeit zurück.
`getUTCMonth`	Gibt den Monat des angegebenen Date-Objektes nach Weltzeit zurück.
`getUTCSeconds`	Gibt die Sekunden des angegebenen Date-Objektes nach Weltzeit zurück.

Methode	Beschreibung
`getYear`	Gibt das Jahr des angegebenen Date-Objektes nach lokaler Zeit zurück.
`setDate`	Setzt den Tag des Monats eines angegebenen Date-Objektes nach lokaler Zeit.
`setFullYear`	Setzt die vollständige Jahreszahl für ein Date-Objekt nach lokaler Zeit.
`setHours`	Setzt die Stunden für ein Date-Objekt nach lokaler Zeit.
`setMilliseconds`	Setzt die Millisekunden für ein Date-Objekt nach lokaler Zeit.
`setMinutes`	Setzt die Minuten für ein Date-Objekt nach lokaler Zeit.
`setMonth`	Setzt den Monat für ein Date-Objekt nach lokaler Zeit.
`setSeconds`	Setzt die Sekunden für ein Date-Objekt nach lokaler Zeit.
`setTime`	Setzt das Datum für das angegebene Date-Objekt in Millisekunden.
`setUTCDate`	Setzt das Datum des angegebenen Date-Objektes nach Weltzeit.
`setUTCFullYear`	Setzt das Jahr des angegebenen Date-Objektes nach Weltzeit.
`setUTCHours`	Setzt die Stunde des angegebenen Date-Objektes nach Weltzeit.
`setUTCMilliseconds`	Setzt die Millisekunden des angegebenen Date-Objektes nach Weltzeit.
`setUTCMinutes`	Setzt die Minute des angegebenen Date-Objektes nach Weltzeit.
`setUTCMonth`	Setzt den durch das angegebene Date-Objekt dargestellten Monat nach Weltzeit.
`setUTCSeconds`	Setzt die Sekunden des angegebenen Date-Objektes nach Weltzeit.
`setYear`	Setzt das Jahr des angegebenen Date-Objektes nach lokaler Zeit.
`toString`	Gibt einen Zeichenfolgenwert mit Datum und Zeit zurück, wie sie im angegebenen Date-Objekt gespeichert sind.
`Date.UTC`	Gibt die Anzahl der Millisekunden zurück, die zwischen Mitternacht des 1. Januars 1970 Weltzeit und der angegebenen Zeit vergangen sind.

Konstruktor für das Date-Objekt

Syntax

```
new Date();
```

```
new Date(jahr [, monat [, datum [, stunde [, minute [, sekunde [,
millisekunde ]]]]]] );
```

Argumente

`jahr` Ein Wert von 0 bis 99 steht für eine Jahreszahl zwischen 1900 und 1999, andernfalls müssen alle vier Ziffern der Jahreszahl angegeben werden.

`monat` Eine Ganzzahl von 0 (Januar) bis 11 (Dezember). Dieses Argument ist optional.

`datum` Eine Ganzzahl von 1 bis 31. Dieses Argument ist optional.

`stunde` Eine Ganzzahl von 0 (Mitternacht) bis 23 (23.00 Uhr).

`minute` Eine Ganzzahl von 0 bis 59. Dieses Argument ist optional.

`sekunde` Eine Ganzzahl von 0 bis 59. Dieses Argument ist optional.

`millisekunde` Eine Ganzzahl von 0 bis 999. Dieses Argument ist optional.

Beschreibung

Objekt; Konstruktor eines neuen Date-Objektes, das das aktuelle Datum und Zeit aufnimmt.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird das aktuelle Datum und die aktuelle Zeit abgerufen.

```
now = new Date();
```

Im folgenden Beispiel wird ein neues Date-Objekt für den Geburtstag von Gary am 7. August 1974 erstellt.

```
gary_birthday = new Date (74, 7, 7);
```

Das folgende Beispiel erstellt ein neues Date-Objekt, verkettet die zurückgegebenen Werte der `getMonth`-, `getDate`- und `getFullYear`-Methoden des Date-Objektes und zeigt diese Werte in dem in der Variablen `dateTextField` angegebenen Textfeld an.

```
myDate = new Date();
dateTextField = (mydate.getMonth() + "/" + myDate.getDate() + "/"
+ mydate.getFullYear());
```

Date.getDate

Syntax
```
meinDatum.getDate();
```

Argumente
Keine.

Beschreibung
Methode; gibt den Tag des Monats (eine Ganzzahl von 1 bis 31) des angegebenen Date-Objektes nach lokaler Zeit zurück.

Player
Flash 5 oder höher.

Date.getDay

Syntax
```
meinDatum.getDay();
```

Argumente
Keine.

Beschreibung
Methode; gibt den Tag des Monats (0 für Sonntag, 1 für Montag usw.) des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Date.getFullYear

Syntax
```
meinDatum.getFullYear();
```

Argumente
Keine.

Beschreibung
Methode; gibt die vollständige Jahreszahl (eine vierstellige Zahl, z. B. 2000) des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird mit dem Konstruktor ein neues Date-Objekt erstellt und der von der `getFullYear`-Methode zurückgegebene Wert an das Ausgabefenster gesendet.

```
myDate = new Date();
trace(myDate.getFullYear());
```

Date.getHours

Syntax
meinDatum`.getHours();`

Argumente
Keine.

Beschreibung
Methode; gibt die Stunde (eine Ganzzahl von 0 bis 23) des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Date.getMilliseconds

Syntax
meinDatum`.getMilliseconds();`

Argumente
Keine.

Beschreibung
Methode; gibt die Millisekunden (eine Ganzzahl von 0 bis 999) des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Date.getMinutes

Syntax
meinDatum`.getMinutes();`

Argumente
Keine.

Beschreibung

Methode; gibt die Minuten (eine Ganzzahl von 0 bis 59) des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player

Flash 5 oder höher.

Date.getMonth

Syntax

```
meinDatum.getMonth();
```

Argumente

Keine.

Beschreibung

Methode; gibt den Monat (0 für Januar, 1 für Februar usw.) des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player

Flash 5 oder höher.

Date.getSeconds

Syntax

```
meinDatum.getSeconds();
```

Argumente

Keine.

Beschreibung

Methode; gibt die Sekunden (eine Ganzzahl von 0 bis 59) des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player

Flash 5 oder höher.

Date.getTime

Syntax

```
meinDatum.getTime();
```

Argumente

Keine.

Beschreibung
Methode; gibt für das angegebene Date-Objekt die Anzahl der Millisekunden (eine Ganzzahl von 0 bis 999) zurück, die seit Mitternacht des 1. Januars 1970 Weltzeit vergangen sind. Mit dieser Methode können Sie bei Vergleichen von zwei oder mehr Zeitangaben, die in unterschiedlichen Zeitzonen definiert sind, einen bestimmten Zeitpunkt darstellen.

Player
Flash 5 oder höher.

Date.getTimezoneOffset

Syntax
```
meinDatum.getTimezoneOffset();
```

Argumente
Keine.

Beschreibung
Methode; gibt die Differenz zwischen der lokalen Zeit des Computers und der Weltzeit in Minuten zurück.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird die Differenz zwischen lokaler Sommerzeit für San Francisco und Weltzeit zurückgegeben. Sommerzeit wird im ausgegebenen Ergebnis nur berücksichtigt, wenn das im Date-Objekt definierte Datum im Zeitraum der Sommerzeit liegt.

```
new Date().getTimezoneOffset();
```

Das Ergebnis lautet wie folgt:

```
420 (7 Stunden * 60 Minuten/Stunde = 420 Minuten)
```

Date.getUTCDate

Syntax
```
meinDatum.getUTCDate();
```

Argumente
Keine.

Beschreibung
Methode; gibt den Tag (das Datum) des Monats des angegebenen Date-Objektes nach Weltzeit zurück.

Player
Flash 5 oder höher.

Date.getUTCDay

Syntax

```
meinDatum.getUTCDate();
```

Argumente

Keine.

Beschreibung

Methode; gibt den Wochentag des angegebenen Date-Objektes nach Weltzeit zurück.

Date.getUTCFullYear

Syntax

```
meinDatum.getUTCFullYear();
```

Argumente

Keine.

Beschreibung

Methode; gibt das vierstellige Jahr des angegebenen Date-Objektes nach Weltzeit zurück.

Player

Flash 5 oder höher.

Date.getUTCHours

Syntax

```
meinDatum.getUTCHours();
```

Argumente

Keine.

Beschreibung

Methode; gibt die Stunden des angegebenen Date-Objektes nach Weltzeit zurück.

Player

Flash 5 oder höher.

Date.getUTCMilliseconds

Syntax

```
meinDatum.getUTCMilliseconds();
```

Argumente

Keine.

Beschreibung
Methode; gibt die Millisekunden des angegebenen Date-Objektes nach Weltzeit zurück.

Player
Flash 5 oder höher.

Date.getUTCMinutes

Syntax
`meinDatum.getUTCMinutes();`

Argumente
Keine.

Beschreibung
Methode; gibt die Minuten des angegebenen Date-Objektes nach Weltzeit zurück.

Player
Flash 5 oder höher.

Date.getUTCMonth

Syntax
`meinDatum.getUTCMonth();`

Argumente
Keine.

Beschreibung
Methode; gibt den Monat des angegebenen Date-Objektes nach Weltzeit zurück.

Player
Flash 5 oder höher.

Date.getUTCSeconds

Syntax
`meinDatum.getUTCSeconds();`

Argumente
Keine.

Beschreibung
Methode; gibt die Sekunden des angegebenen Date-Objektes nach Weltzeit zurück.

Player
Flash 5 oder höher.

Date.getYear

Syntax
`meinDatum.getYear();`

Argumente
Keine.

Beschreibung
Methode; gibt das Jahr des angegebenen Date-Objektes nach lokaler Zeit zurück. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird. Das Jahr ist die vollständige Jahreszahl minus 1900. Das Jahr 2000 wird z. B. als 100 dargestellt.

Player
Flash 5 oder höher.

Date.setDate

Syntax
`meinDatum.setDate(datum);`

Argumente
`datum` Eine Ganzzahl von 1 bis 31.

Beschreibung
Methode; setzt den Tag des Monats für das angegebene Date-Objekt nach lokaler Zeit. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Date.setFullYear

Syntax
`meinDatum.setFullYear(jahr [, monat [, datum]] );`

Argumente
`jahr` Eine vierstellige Zahl zur Angabe eines Jahres. Zweistellige Zahlen sind nicht für Jahresangabe vorgesehen; 99 steht z. B. nicht für das Jahr 1999, sondern für das Jahr 99.

monat Eine Ganzzahl von 0 (Januar) bis 11 (Dezember). Dieses Argument ist optional.

datum Eine Zahl von 1 bis 31. Dieses Argument ist optional.

Beschreibung
Methode; setzt das Jahr des angegebenen Date-Objektes nach lokaler Zeit. Wenn die Argumente *monat* und *datum* angegeben werden, werden sie auf lokale Zeit gesetzt. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Die Ergebnisse von `getUTCDay` und `getDay` können sich durch Aufrufen dieser Methode ändern.

Player
Flash 5 oder höher.

Date.setHours

Syntax
```
meinDatum.setHours(stunde);
```

Argumente
stunde Eine Ganzzahl von 0 (Mitternacht) bis 23 (23.00 Uhr).

Beschreibung
Methode; setzt die Stunden des angegebenen Date-Objektes nach lokaler Zeit. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Date.setMilliseconds

Syntax
```
meinDatum.setMilliseconds(millisekunde);
```

Argumente
millisekunde Eine Ganzzahl von 0 bis 999.

Beschreibung
Methode; setzt die Millisekunden des angegebenen Date-Objektes nach lokaler Zeit. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Date.setMinutes

Syntax

`meinDatum.setMinutes(minute);`

Argumente

minute Eine Ganzzahl von 0 bis 59.

Beschreibung

Methode; setzt die Minuten des angegebenen Date-Objektes nach lokaler Zeit. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player

Flash 5 oder höher.

Date.setMonth

Syntax

`meinDatum.setMonth(monat [, datum ]);`

Argumente

monat Eine Ganzzahl von 0 (Januar) bis 11 (Dezember).

datum Eine Ganzzahl von 1 bis 31. Dieses Argument ist optional.

Beschreibung

Methode; setzt den Monat des angegebenen Date-Objektes nach lokaler Zeit. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player

Flash 5 oder höher.

Date.setSeconds

Syntax

`meinDatum.setSeconds(sekunde);`

Argumente

sekunde Eine Ganzzahl von 0 bis 59.

Beschreibung

Methode; setzt die Sekunden für das angegebene Date-Objekt nach lokaler Zeit. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player

Flash 5 oder höher.

Date.setTime

Syntax
`meinDatum.setTime(millisekunde);`

Argumente
millisekunde Eine Ganzzahl von 0 bis 999.

Beschreibung
Methode; setzt das Datum für das angegebene Date-Objekt in Millisekunden.

Player
Flash 5 oder höher.

Date.setUTCDate

Syntax
`meinDatum.setUTCDate(datum);`

Argumente
datum Eine Ganzzahl von 1 bis 31.

Beschreibung
Methode; setzt das Datum für das angegebene Date-Objekt nach Weltzeit. Durch Aufrufen dieser Methode werden die anderen Felder des angegebenen Datums nicht verändert. Die `getUTCDay`- und `getDay`-Methoden können jedoch einen neuen Wert zurückgeben, wenn der Wochentag aufgrund des Aufrufs dieser Methode geändert wird.

Player
Flash 5 oder höher.

Date.setUTCFullYear

Syntax
`meinDatum.setUTCFullYear(jahr [, monat [, datum]])`

Argumente
jahr Das als vierstellige Zahl angegebene Jahr, z.B. 2000.

monat Eine Ganzzahl von 0 (Januar) bis 11 (Dezember). Dieses Argument ist optional.

datum Eine Ganzzahl von 1 bis 31. Dieses Argument ist optional.

Beschreibung
Methode; setzt das Jahr des angegebenen Date-Objektes (*meinDatum*) nach Weltzeit.

Optional kann diese Methode auch Monat und Datum setzen, die durch das angegebene Date-Objekt dargestellt werden. Dabei werden keine weiteren Felder des Date-Objektes geändert. Durch Aufrufen von `setUTCFullyear` kann `getUTCDay` und `getDay` einen neuen Wert zurückgeben, wenn sich aufgrund dieser Operation der Wochentag ändert.

Player
Flash 5 oder höher.

Date.setUTCHours

Syntax
```
meinDatum.setUTCHours(stunde [, minute [, sekunde [, mil-
lisekunde]]]));
```

Argumente

stunde Eine Ganzzahl von 0 (Mitternacht) bis 23 (23.00 Uhr).

minute Eine Ganzzahl von 0 bis 59. Dieses Argument ist optional.

sekunde Eine Ganzzahl von 0 bis 59. Dieses Argument ist optional.

millisekunde Eine Ganzzahl von 0 bis 999. Dieses Argument ist optional.

Beschreibung
Methode; setzt die Stunde für das angegebene Date-Objekt nach Weltzeit.

Player
Flash 5 oder höher.

Date.setUTCMilliseconds

Syntax
```
meinDatum.setUTCMilliseconds(millisekunde);
```

Argumente

millisekunde Eine Ganzzahl von 0 bis 999.

Beschreibung
Methode; setzt die Millisekunden für das angegebene Date-Objekt nach Weltzeit.

Player
Flash 5 oder höher.

Date.setUTCMinutes

Syntax

`meinDatum.setUTCMinutes(minute [, sekunde [, millisekunde]]));`

Argumente

minute Eine Ganzzahl von 0 bis 59.

sekunde Eine Ganzzahl von 0 bis 59. Dieses Argument ist optional.

millisekunde Eine Ganzzahl von 0 bis 999. Dieses Argument ist optional.

Beschreibung

Methode; setzt die Minute für das angegebene Date-Objekt nach Weltzeit.

Player

Flash 5 oder höher.

Date.setUTCMonth

Syntax

`meinDatum.setUTCMonth(monat [, datum]);`

Argumente

monat Eine Ganzzahl von 0 (Januar) bis 11 (Dezember).

datum Eine Ganzzahl von 1 bis 31. Dieses Argument ist optional.

Beschreibung

Methode; setzt den Monat und optional den Tag (das Datum) für das angegebene Date-Objekt nach Weltzeit. Durch Aufrufen dieser Methode werden die anderen Felder des angegebenen Date-Objektes nicht verändert. Die Methoden `getUTCDay` und `getDay` können jedoch einen neuen Wert anzeigen, wenn sich der Wochentag aufgrund der Angabe des Arguments *datum* beim Aufruf von `setUTCMonth` ändert.

Player

Flash 5 oder höher.

Date.setUTCSeconds

Syntax

`meinDatum.setUTCSeconds(sekunde [, millisekunde]));`

Argumente

sekunde Eine Ganzzahl von 0 bis 59.

millisekunde Eine Ganzzahl von 0 bis 999. Dieses Argument ist optional.

Beschreibung
Methode; setzt die Sekunden für das angegebene Date-Objekt nach Weltzeit.

Player
Flash 5 oder höher.

Date.setYear

Syntax
```
meinDatum.setYear(jahr);
```

Argumente
jahr Eine vierstellige Zahl, z.B. 2000.

Beschreibung
Methode; setzt das Jahr für das angegebene Date-Objekt nach lokaler Zeit. Die lokale Zeit wird durch das Betriebssystem bestimmt, auf dem der Flash Player ausgeführt wird.

Player
Flash 5 oder höher.

Date.toString

Syntax
```
meinDatum.toString();
```

Argumente
Keine.

Beschreibung
Methode; gibt für das angegebene Date-Objekt einen Zeichenfolgenwert in lesbarem Format zurück.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel werden die Informationen im Date-Objekt dateOfBirth als Zeichenfolge zurückgegeben.

```
var dateOfBirth = newDate(74, 7, 7, 18, 15);
trace (dateOfBirth.toString());
```

Ausgabe (für Pazifik-Normalzeit):

```
Wed Aug 7 18:15:00 GMT-0700 1974
```

Date.UTC

Syntax

```
Date.UTC(jahr, monat [, datum [, stunde [, minute [, sekunde [,
millisekunde ]]]]]);
```

Argumente

jahr Eine vierstellige Zahl, z.B. 2000.

monat Eine Ganzzahl von 0 (Januar) bis 11 (Dezember).

datum Eine Ganzzahl von 1 bis 31. Dieses Argument ist optional.

stunde Eine Ganzzahl von 0 (Mitternacht) bis 23 (23.00 Uhr).

minute Eine Ganzzahl von 0 bis 59. Dieses Argument ist optional.

sekunde Eine Ganzzahl von 0 bis 59. Dieses Argument ist optional.

millisekunde Eine Ganzzahl von 0 bis 999. Dieses Argument ist optional.

Beschreibung

Methode; gibt die Anzahl der Millisekunden zurück, die seit Mitternacht des 1. Januar 1970 Weltzeit und der in den Argumenten angegebenen Zeit vergangen sind. Dies ist eine statische Methode, die nicht durch ein bestimmtes Date-Objekt, sondern durch den Konstruktor des Date-Objektes aufgerufen wird. Mit dieser Methode können Sie ein Date-Objekt erstellen, das Weltzeit zugrunde legt. Der Date-Konstruktor geht hingegen von lokaler Zeit aus.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird ein neues Date-Objekt `gary_birthday` erstellt, das in Weltzeit definiert ist. Dies ist die Weltzeit-Variante des Beispiels für die Konstruktormethode `new Date()`.

```
gary_birthday = new Date(Date.UTC(1974, 7, 8));
```

delete

Syntax

```
delete (referenz);
```

Argumente

referenz Der Name der zu löschenden Variablen bzw. des zu löschenden Objektes.

Beschreibung

Operator; zerstört das als *referenz* angegebene Objekt bzw. die Variable und gibt `true` zurück, wenn das Objekt erfolgreich gelöscht wurde; andernfalls wird `false` zurückgegeben. Dieser Operator ist nützlich, um von Skripten verwendeten Speicher freizugeben. Obwohl `delete` ein Operator ist, wird er in der Regel als Anweisung verwendet:

```
delete x;
```

Der Delete-Operator bricht den Vorgang möglicherweise ab und gibt false zurück, wenn *referenz* nicht vorhanden ist oder nicht gelöscht werden darf. Vordefinierte Objekte und Eigenschaften sowie mit `var` definierte Variablen dürfen nicht gelöscht werden.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird ein Objekt erstellt, verwendet und gelöscht, sobald es nicht mehr benötigt wird.

```
account = new Object();
  account.name = 'Jon';
  account.balance = 10000;
  ...
  delete account;
```

Im folgenden Beispiel wird eine Objekteigenschaft gelöscht.

```
// Erstellen des neuen Objektes "account"
account = new Object();
// Zuweisen des Eigenschaftsnamen zum Konto
  account.name = 'Jon';
// Löschen der Eigenschaft
delete account.name;
```

Ein weiteres Beispiel zum Löschen einer Objekteigenschaft lautet wie folgt:

```
// Erstellen eines Array-Objektes mit der Länge 0
array = new Array();
// Array.length ist jetzt 1
   array[0] = "abc";
// Hinzufügen eines weiteren Elements hinzufügen zum Array,
Array.length ist jetzt 2
   array[1] = "def";
// Hinzufügen eines weiteren Elements hinzufügen zum Array,
Array.length ist jetzt 3
   array[2] = "ghi";
// array[2] wird gelöscht, aber Array.length wird nicht geändert,
   delete array[2];
```

Im folgenden Beispiel wird das Verhalten von `delete` bei Objektreferenzen demonstriert.

```
// Erstellen eines neuen Objektes und Zuweisen der Variablen ref1
als Referenz auf das Objekt
ref1 = new Object();
ref1.name = "Jody";
// Kopieren der Referenzvariablen in eine neue Variable und
Löschen von ref1
ref2 = ref1;
delete ref1;
```

Ohne das Kopieren von `ref1` in `ref2` würde beim Löschen von `ref1` das Objekt gelöscht werden, da keine Referenzen auf dieses Objekt vorhanden wären. Nach dem Löschen von `ref2` wären keine Referenzen mehr auf das Objekt vorhanden. Das Objekt würde zerstört und der von ihm verwendete Speicher freigegeben.

Siehe auch

„var" auf Seite 381

do... while

Syntax

```
do {
anweisung;
} while (bedingung);
```

Argumente

bedingung Die auszuwertende Bedingung.

anweisung Die Anweisung, die ausgeführt werden soll, solange die *bedingung* den Wert `true` hat.

Beschreibung

Aktion; führt die Anweisungen aus und wertet die Bedingung dann in einer Schleife aus, solange die Bedingung den Wert true hat.

Player

Flash 4 oder höher.

Siehe auch

„break" auf Seite 234
„continue" auf Seite 239

_droptarget

Syntax

ziehenInstanzname._droptarget

Argumente

ziehenInstanzname Der Name einer Filmsequenzinstanz, die Ziel einer startDrag-Aktion war.

Beschreibung

Eigenschaft (schreibgeschützt); gibt den absoluten Pfad der Filmsequenzinstanz in Schrägstrich-Syntax zurück, auf die *ziehenInstanzname*verschoben wurde. Die _droptarget-Eigenschaft gibt immer einen Pfad zurück, der mit / beginnt. Um die _droptarget-Eigenschaft einer Instanz mit einer Referenz zu vergleichen, wandeln Sie den zurückgegebenen Wert in Schrägstrich-Syntax mit eval in eine Referenz um.

Player

Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird die _droptarget-Eigenschaft der Filmsequenzinstanz garbage ausgewertet und mit eval von Schrägstrich-Syntax in eine Referenz in Punkt-Syntax umgewandelt. Die Referenz garbage wird dann mit der Referenz auf die Filmsequenzinstanz trash verglichen. Wenn die beiden Referenzen äquivalent sind, wird die Sichtbarkeit von garbage auf false gesetzt. Wenn sie nicht äquivalent sind, wird die Instanz garbage auf die Ausgangsposition zurückgesetzt.

```
if (eval(garbage._droptarget) == _root.trash) {
   garbage._visible = false;
} else {
   garbage._x = x_pos;
   garbage._y = y_pos;
}
```

Die Variablen x_pos und y_pos werden in Bild 1 des Filmes mit dem folgenden Skript gesetzt:

```
x_pos = garbage._x;
y_pos = garbage._y;
```

Siehe auch

„startDrag“ auf Seite 362

duplicateMovieClip

Syntax

`duplicateMovieClip(ziel, neuerName, tiefe);`

Argumente

`ziel` Der Zielpfad des zu duplizierenden Filmes.

`neuerName` Ein eindeutiger Bezeichner für die duplizierte Filmsequenz.

`tiefe` Die Tiefenebene der Filmsequenz. Die Tiefenebene gibt die Position in der Stapelreihenfolge an, von der abhängt, wie überlappende Filmsequenzen und andere Objekte angezeigt werden. Der ersten erstellten Filmsequenz oder der auf die Bühne gezogenen Instanz wird die Tiefenebene 0 zugewiesen. Jeder folgenden oder duplizierten Filmsequenz muss eine andere Tiefenebene zugewiesen werden, um zu verhindern, dass Filme auf bereits verwendeten Ebenen oder die ursprüngliche Filmsequenz ersetzt werden.

Beschreibung

Aktion; erstellt während der Wiedergabe des Filmes eine Instanz einer Filmsequenz. Duplizierte Filmsequenzen starten immer bei Bild 1 unabhängig von dem Bild, an dem sich die Originalfilmsequenz befindet. In der übergeordneten Filmsequenz enthaltene Variablen werden nicht in die duplizierte Filmsequenz kopiert. Beim Löschen der übergeordneten Filmsequenz wird auch die duplizierte Filmsequenz gelöscht. Mit der Aktion oder Methode `removeMovieClip` können Sie eine Filmsequenzinstanz löschen, die mit `duplicateMovieClip` erstellt wurde.

Player

Flash 4 oder höher.

Beispiel

Diese Anweisung dupliziert die Filmsequenzinstanz `flower` zehnmal. Mit der Variablen `i` wird ein neuer Instanzname und eine Tiefe erstellt.

```
on(release) {
   amount = 10;
   while(amount>0) {
         duplicateMovieClip (_root.flower, "mc" + i, i);
         setProperty("mc" + i, _x, random(275));
         setProperty("mc" + i, _y, random(275));
         setProperty("mc" + i, _alpha, random(275));
         setProperty("mc" + i, _xscale, random(50));
         setProperty("mc" + i, _yscale, random(50));
         i = i + 1;
         amount = amount-1;
    }
}
```

Siehe auch

„`removeMovieClip`“ auf Seite 346
„`MovieClip.removeMovieClip`“ auf Seite 320

else {

Syntax

`else {anweisung(en)};`

Argumente

`anweisung(en)` Eine alternative Reihe von Anweisungen, die ausgeführt werden, wenn die in der `if`-Anweisung angegebene Bedingung `false` ist.

Beschreibung

Aktion; gibt die Aktionen, Anweisungen, Argumente oder andere Bedingungen an, die ausgeführt werden, wenn die erste `if`-Anweisung `false` zurückgibt.

Player

Flash 4 oder höher.

Siehe auch

„if“ auf Seite 276

eq (gleich - zeichenfolgenspezifisch)

Syntax

`ausdruck1` `eq` *`ausdruck2`*

Argumente

`ausdruck1, ausdruck2` Zahlen, Zeichenfolgen oder Variablen.

Beschreibung

Vergleichsoperator; vergleicht zwei Ausdrücke auf Gleichheit und gibt `true` zurück, wenn *`ausdruck1`* gleich *`ausdruck2`* ist; andernfalls wird `false` zurückgegeben.

Player

Flash 1 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen ==-Operators (Gleichheit) empfohlen.

Siehe auch

„== (gleich)“ auf Seite 215

escape

Syntax

`escape(`*`ausdruck`*`);`

Argumente

`ausdruck` Der Ausdruck, der in eine Zeichenfolge umgewandelt und URL-kodiert werden soll.

Beschreibung
Funktion; wandelt das Argument in eine Zeichenfolge um und kodiert es in URL-Format, in dem alle alphanumerischen Zeichen mit % gefolgt von einer hexadezimalen Sequenz kodiert werden.

Player
Flash 5 oder höher.

Beispiel
```
escape("Hello{[World]}„);
```
Das Ergebnis des genannten Codes lautet wie folgt:

```
Hello%7B%5BWorld%5D%7D
```

Siehe auch
„unescape“ auf Seite 379

eval

Syntax
```
eval(ausdruck);
```

Argumente
ausdruck Eine Zeichenfolge, die den Namen von abzurufenden Variablen, Eigenschaften, Objekten oder Filmsequenzen enthält.

Beschreibung
Funktion; greift dem Namen nach auf Variablen, Eigenschaften, Objekte oder Filmsequenzen zu. Wenn *ausdruck* eine Variable oder eine Eigenschaft ist, wird der Wert der Variablen oder Eigenschaft zurückgegeben. Wenn *ausdruck* ein Objekt oder eine Filmsequenz ist, wird eine Referenz auf das Objekt bzw. die Filmsequenz zurückgegeben. Wenn das in *ausdruck* angegebene Element nicht gefunden werden kann, wird undefiniert zurückgegeben.

In Flash 4 konnten mit der `eval`-Funktion Arrays simuliert werden. In Flash 5 wird die Verwendung des Array-Objektes zum Erstellen von Arrays empfohlen.

Anmerkung: Die `eval`-Aktion in ActionScript stimmt nicht mit der `eval`-Funktion in JavaScript überein und kann nicht zum Auswerten von Anweisungen eingesetzt werden.

Player
Voller Funktionsumfang in Flash 5 oder höher. Sie können `eval` beim Exportieren in den Flash 4 Player einsetzen. Sie müssen jedoch die Schrägstrich-Syntax verwenden und können nur auf Variablen, nicht aber auf Eigenschaften oder Objekte zugreifen.

Beispiel
Im folgenden Beispiel wird mit `eval` der Wert der Variablen `x` bestimmt und auf den Wert von `y` gesetzt.

```
x = 3;
y = eval("x");
```

Im folgenden Beispiel wird mit eval auf das einer Filmsequenzinstanz auf der Bühne zugeordnete Filmsequenzobjekt `Ball` verwiesen.

```
eval("_root.Ball");
```

Siehe auch
„`Array (Objekt)`“ auf Seite 222

evaluate

Syntax
`anweisung`;

Argumente
Keine.

Beschreibung
Aktion; erstellt eine neue leere Zeile und fügt ein `;` zur Eingabe von eindeutigen Skriptanweisungen mit dem Feld „Ausdruck“ im Bedienfeld „Aktionen“ ein. Mit der `evaluate`-Anweisung können Benutzer, die im normalen Modus des Bedienfeldes „Aktionen“ von Flash 5 Skripte erstellen, Funktionen aufrufen.

Player
Flash 5 oder höher.

_focusrect

Syntax
`_focusrect = `*`Boolean`*;

Argumente
`Boolean` `true` oder `false`.

Beschreibung
Eigenschaft (global); gibt an, ob um die momentan markierte Schaltfläche ein gelbes Rechteck angezeigt wird. Mit dem Standardwert `true` (nicht-Null) wird dem Benutzer beim Navigieren durch Drücken der TAB-Taste ein gelbes Rechteck um die aktuell markierte Schaltfläche oder das Textfeld angezeigt. Geben Sie `false` an, wenn bei der Navigation durch den Benutzer nur der Schaltflächenstatus „over“ (sofern definiert) angezeigt werden soll.

Player
Flash 4 oder höher.

for

Syntax

```
for(init; bedingung; nächster); {
anweisung;
}
```

Argumente

init Ein Ausdruck, der vor Beginn der Schleifensequenz ausgewertet werden muss, in der Regel ein Zuweisungsausdruck. Eine `var`-Anweisung ist für dieses Argument ebenfalls zulässig.

bedingung Eine Bedingung, die den Wert `true` oder `false` haben kann. Die Bedingung wird vor jeder Schleifenwiederholung ausgewertet; die Schleife wird beendet, wenn die Bedingung den Wert `false` ergibt.

nächster Ein Ausdruck, der nach jeder Schleifenwiederholung ausgewertet wird; in der Regel ein Zuweisungsausdruck, der die Operatoren `++` (Inkrement) oder `--` (Dekrement) verwendet.

anweisung Eine Anweisung innerhalb des Schleifenabschnitts, die ausgeführt werden soll.

Beschreibung

Aktion; ein Schleifenkonstrukt, das den Ausdruck `init` (initialisieren) einmal auswertet und dann eine Schleife startet, in der die *anweisung* ausgeführt und der nächste Ausdruck ausgewertet wird, solange die *bedingung* den Wert `true` hat.

Einige Eigenschaften können in den Aktionen `for` oder `for..in` nicht gezählt werden. Beispielsweise sind die integrierten Methoden des Array-Objektes (`Array.sort` und `Array.reverse`) nicht in der Aufzählung eines Array-Objektes enthalten. Eigenschaften von Filmsequenzen wie `_x` und `_y` sind nicht zählbar.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird `for` zum Addieren der Elemente eines Arrays verwendet.

```
for(i=0; i<10; i++) {
array [i] = (i + 5)*10;
}
```

Das folgende Array wird zurückgegeben:

```
[50, 60, 70, 80, 90, 100, 110, 120, 130, 140]
```

Im folgenden Beispiel wird mit `for` die gleiche Aktion wiederholt durchgeführt. Im folgenden Code addiert die `for`-Schleife die Zahlen von 1 bis 100.

```
var sum = 0;
  for (var i=1; i<=100; i++) {
    sum = sum + i;
  }
```

Siehe auch

„++ (Inkrement)“ auf Seite 189
„-- (Dekrement)“ auf Seite 189
„for..in“ auf Seite 267
„var“ auf Seite 381

for..in

Syntax

```
for(iterationVariablein objekt){
anweisung; }
```

Argumente

iterationVariable Der Name einer Variablen, die als Iterator fungiert, indem sie auf jede Objekteigenschaft oder jedes Arrayelement verweist.

objekt Der Name des Objektes, das durchlaufen werden soll.

anweisung Eine Anweisung, die für jede Iteration ausgeführt wird.

Beschreibung

Aktion; führt eine Schleife durch die Eigenschaften eines Objektes oder die Elemente eines Arrays aus und führt die *anweisung* für jede Objekteigenschaft aus.

Einige Eigenschaften können in den Aktionen `for` oder `for..in` nicht gezählt werden. Beispielsweise sind die integrierten Methoden des Array-Objektes (`Array.sort` und `Array.reverse`) nicht in der Aufzählung eines Array-Objektes enthalten. Eigenschaften von Filmsequenzen, beispielsweise `_x` und `_y`, sind nicht zählbar.

Das `for..in`-Konstrukt durchläuft alle Objekteigenschaften in der Prototyp-Kette des durchlaufenen Objektes. Wenn der Prototyp dem Unterelement übergeordnet ist, werden bei Durchlaufen der Eigenschaften des Unterelements mit `for...in` auch die Eigenschaften des übergeordneten Elements durchlaufen.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel werden mit „for..in“ die Eigenschaften eines Objektes durchlaufen.

```
myObject = { name:'Tara', age:27, city:'San Francisco' };
for (name in myObject) {
  trace ("myObject." + name + " = " + myObject[name]);
}
```

Die Ausgabe dieses Beispiels lautet wie folgt:

```
myObject.name = Tara
myObject.age = 27
myObject.city = San Francisco
```

Im folgenden Beispiel wird der typeof-Operator mit „for..in“ verwendet, um einen bestimmten Typ eines Unterelements zu durchlaufen.

```
for (name in myMovieClip) {
  if (typeof (myMovieClip[name]) = "movieclip") {
    trace ("Es gibt eine untergeordnete Filmsequenz namens " +
name);
  }
}
```

Im folgenden Beispiel werden die Unterelemente einer Filmsequenz aufgezählt und jeweils in den entsprechenden Zeitleisten zu Bild 2 gesendet. Die Filmsequenz RadioButtonGroup ist ein übergeordnetes Element mit verschiedenen Unterelementen, _RedRadioButton_, _GreenRadioButton_ und_BlueRadioButton.

```
for (var name in RadioButtonGroup) {
  RadioButtonGroup[name].gotoAndStop(2);
}
```

_framesloaded

Syntax

instanzname._framesloaded

Argumente

instanzname Der Name der auszuwertenden Filmsequenzinstanz.

Beschreibung

Eigenschaft (schreibgeschützt); die Anzahl der Bilder, die aus einem Streaming-Film geladen wurden. Diese Eigenschaft ist nützlich, um festzustellen, ob die Inhalte eines bestimmen Bilds und aller vorausgehenden Bilder geladen wurden und im Browser eines Benutzers lokal zur Verfügung stehen. Diese Eigenschaft dient außerdem der Überwachung des Downloadvorgangs großer Filme. Sie könnten z.’B. eine Meldung anzeigen lassen, dass der Film geladen wird, bis ein bestimmtes Bild in dem Film vollständig geladen wurde.

Player

Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird mit Hilfe der `_framesloaded` -Eigenschaft der Start eines Filmes mit der Anzahl der geladenen Bilder verknüpft.

```
if (_framesloaded >= _totalframes) {
gotoAndPlay ("Scene 1", "start");
} else {
setProperty ("_root.loader", _xscale, (_framesloaded/
_totalframes)*100);
}
```

fscommand

Syntax

```
fscommand(befehl, argumente);
```

Argumente

befehl Eine Zeichenfolge, die zur beliebigen weiteren Verwendung an die Host-Anwendung übergeben wird.

argumente Eine Zeichenfolge, die zur beliebigen weiteren Verwendung an die Host-Anwendung übergeben wird.

Beschreibung

Aktion; ermöglicht die Kommunikation zwischen dem Flash Film und dem Programm, das den Flash Player ausführt. In einem Webbrowser ruft `fscommand` die JavaScript-Funktion `filmname_Dofscommand` auf der HTML-Seite auf, die den Flash Film enthält. Dabei steht `filmname` für den Namen des Flash Player, wie er durch das Attribut `NAME` des `EMBED`-Tags oder durch die ID-Eigenschaft des `OBJECT`-Tags zugewiesen wurde. Wenn Flash Player den Namen `Film01` erhalten hat, dann lautet die aufgerufene JavaScript-Funktion `Film01_DoFSCommand`.

Player

Flash 3 oder höher.

function

Syntax

```
function funktionsname ([ argument0, argument1,...argumentN]){
anweisung(en)
}
```

```
function ([argument0, argument1,...argumentN]){
anweisung(en)
}
```

Argumente

funktionsname Der Name der neuen Funktion.

argument Keine, eine oder mehr Zeichenfolgen, Zahlen oder Objekte, die an `function` übergeben werden.

anweisungen Keine, eine oder mehr ActionScript-Anweisungen, die für den Hauptabschnitt der `function` definiert wurden.

Beschreibung

Aktion; eine Gruppe von Anweisungen, die zur Ausführung einer bestimmten Aufgabe definiert werden. Sie können eine Funktion an einer Stelle *deklarieren* oder definieren und diese aus verschiedenen Skripts in einem Film aufrufen. Beim Definieren einer Funktion können Sie auch Argumente für diese Funktion angeben. Argumente sind Platzhalter für Werte, die von der Funktion für die Ausführung verwendet werden. Sie können einer Funktion bei jedem Aufruf unterschiedliche Argumente, auch Parameter genannt, übergeben.

Mit der `return`-Aktion in den *`anweisung(en)`* einer Funktion weisen Sie die Funktion an, einen Wert zurückzugeben oder zu generieren.

Verwendung 1: Deklariert eine `function` mit dem angegebenen *`funktionsnamen`*, den *`argumenten`* und *`anweisung(en)`*. Beim Aufrufen einer Funktion wird die Funktionsdeklaration aufgerufen. Vorwärtsreferenzen sind zulässig; eine Funktion kann innerhalb derselben Aktionsliste nach einem Aufruf deklariert werden. Eine Funktionsdeklaration ersetzt alle früheren Deklarationen derselben Funktion. Sie können diese Syntax an jeder Stelle verwenden, an der eine Anweisung zulässig ist.

Verwendung 2: Erstellt eine anonyme Funktion und gibt diese zurück. Diese Syntax wird in Ausdrücken verwendet und ist besonders nützlich für das Installieren von Methoden in Objekten.

Player

Flash 5 oder höher.

Beispiel

(Verwendung 1) Im folgenden Beispiel wird die Funktion `sqr` definiert, die ein Argument entgegennimmt und das Quadrat `(x*x)` des Arguments zurückgibt. Beachten Sie, dass bei Deklaration und Verwendung einer Funktion in demselben Skript die Funktionsdeklaration sich hinter der Verwendung der Funktion befinden kann.

```
y=sqr(3);
function sqr(x) {
return x*x;
}
```

(Verwendung 2) Die folgende Funktion definiert ein Circle-Objekt:

```
function Circle(radius) {
  this.radius = radius;
}
```

Die folgende Anweisung definiert eine anonyme Funktion, die den Flächeninhalt eines Kreises berechnet und sie als Methode mit dem `Circle`-Objekt verbindet:

```
Circle.prototype.area = function () {return Math.PI * this.radius
* this.radius}
```

ge (größer oder gleich - zeichenfolgenspezifisch)

Syntax

ausdruck1 ge *ausdruck2*

Argumente

ausdruck1, ausdruck2 Zahlen, Zeichenfolgen oder Variablen.

Beschreibung

Operator (Vergleich); vergleicht *ausdruck1* mit *ausdruck2* und gibt `true` zurück, wenn *ausdruck1* größer oder gleich *ausdruck2* ist; andernfalls wird `false` zurückgegeben.

Player

Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen >=-Operators empfohlen.

Siehe auch

„>= (größer oder gleich)“ auf Seite 216

getProperty

Syntax

getProperty(*instanzname, eigenschaft*);

Argumente

instanzname Der Instanzname einer Filmsequenz, für die die Eigenschaft abgerufen wird.

eigenschaft Eine Eigenschaft einer Filmsequenz, beispielsweise eine *x*- oder *y*-Koordinate.

Beschreibung

Funktion; gibt den Wert der angegebenen *eigenschaft* für die Filmsequenz-instanz zurück.

Player

Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird die Koordinate auf der horizontalen Achse (`_x`) für die Filmsequenz `myMovie` abgerufen.

```
getProperty(_root.myMovie_item._x);
```

getTimer

Syntax
`getTimer();`

Argumente
Keine.

Beschreibung
Funktion; gibt die Anzahl der Millisekunden zurück, die seit dem Beginn der Wiedergabe des Filmes vergangen sind.

Player
Flash 4 oder höher.

getURL

Syntax
`getURL(`*`url`*` [, `*`fenster`*`[, `*`variablen`*`]]);`

Argumente
`url` Der URL, von dem das Dokument geladen wird. Der URL muss sich in der gleichen Subdomäne befinden wie der URL, unter dem der Film momentan abgelegt ist.

`fenster` Ein optionales Argument, das das Fenster oder den HTML-Frame angibt, in dem das Dokument geladen werden soll. Geben Sie den Namen eines bestimmten Fensters ein oder wählen Sie aus folgenden reservierten Zielnamen aus:

- `_self` bezeichnet den aktuellen Frame im aktuellen Fenster.
- `_blank` bezeichnet ein neues Fenster.
- `_parent` bezeichnet den übergeordneten Frame des aktuellen Rahmens.
- `_top` bezeichnet den obersten Frame im aktuellen Fenster.

`variablen` Ein optionales Argument, das eine Methode für das Senden von Variablen angibt. Wenn es keine Variablen gibt, lassen Sie dieses Argument aus; andernfalls geben Sie mit Hilfe der `GET`- oder `POST`-Methode an, ob Variablen geladen werden sollen. Bei `GET` werden die Variablen am Ende des URL angehängt. `GET` wird bei einer geringen Anzahl von Variablen verwendet. Bei `POST` werden die Variablen in einer separaten HTTP-Kopfzeile gesendet. `POST` wird bei langen Zeichenfolgen für Variablen verwendet.

Beschreibung

Aktion; lädt ein Dokument von einem bestimmten URL in ein Fenster oder übergibt Variablen an eine andere Anwendung unter einem definierten URL. Damit Sie diese Aktion testen können, muss die zu ladende Datei am angegebenen Ort gespeichert sein. Für die Verwendung eines absoluten URL (z.B. *`http://www.meinserver.de`*) ist eine Netzwerkverbindung erforderlich.

Player

Flash 2 oder höher. Die Optionen GET und POST sind nur für Flash 4 Player und höhere Versionen verfügbar.

Beispiel

In diesem Beispiel wird ein neuer URL in ein leeres Browserfenster geladen. Die `getURL`-Aktion gibt die Variable `incomingAd` als den *`url`*-Parameter an, so dass Sie den geladenen URL ändern können, ohne den Flash Film bearbeiten zu müssen. Der Wert der Variablen `incomingAd` wird Flash zu einem früheren Zeitpunkt des Filmes mit einer `loadVariables`-Aktion übergeben.

```
on(release) {
    getURL(incomingAd, "_blank");
}
```

Siehe auch

„`loadVariables`“ auf Seite 292
„`XML.send`“ auf Seite 402
„`XML.sendAndLoad`“ auf Seite 403
„`XMLSocket.send`“ auf Seite 412

getVersion

Syntax

```
getVersion();
```

Argumente

Keine.

Beschreibung

Funktion; gibt eine Zeichenfolge zurück, die Flash Player Versions- und Plattforminformationen enthält.

Diese Funktion funktioniert nicht im Filmtestmodus und gibt nur Informationen über Flash Player 5 und höhere Versionen zurück.

Beispiel

Ein Beispiel für eine Zeichenfolge, die von der `getVersion`-Funktion zurückgegeben wurde, lautet wie folgt:

```
WIN 5,0,17,0
```

Darin wird angegeben, dass es sich bei der Plattform um Windows handelt und die Versionsnummer des Flash Player Hauptversion 5, Unterversion 17(5.0r17) ist.

Player
Flash 5 oder höher.

gotoAndPlay

Syntax
`gotoAndPlay(szene, bild);`

Argumente
szene Der Szenenname, zu dem der Abspielkopf gesendet wird.

bild Die Nummer des Bilds, zu dem der Abspielkopf gesendet wird.

Beschreibung
Aktion; sendet den Abspielkopf zum angegebenen Bild in einer Szene und startet den Abspielvorgang von diesem Bild aus. Wenn keine Szene angegeben ist, springt der Abspielkopf zum angegebenen Bild in der aktuellen Szene.

Player
Flash 2 oder höher.

Beispiel
Wenn der Benutzer auf eine Schaltfläche klickt, der die `gotoAndPlay`-Aktion zugeordnet ist, wird der Abspielkopf zu Bild 16 verschoben und beginnt dort die Wiedergabe.

```
on(release) {
    gotoAndPlay(16);
}
```

gotoAndStop

Syntax
`gotoAndStop(szene, bild);`

Argumente
szene Der Szenenname, zu dem der Abspielkopf gesendet wird.

bild Die Nummer des Bilds, zu dem der Abspielkopf gesendet wird.

Beschreibung
Aktion; sendet den Abspielkopf zum angegebenen Bild in einer Szene und hält den Film dort an. Wenn keine Szene angegeben ist, springt der Abspielkopf zum Bild in der aktuellen Szene.

Player
Flash 2 oder höher.

Beispiel

Wenn der Benutzer auf eine Schaltfläche klickt, der die `gotoAndStop`-Aktion zugeordnet ist, wird der Abspielkopf zu Bild 5 verschoben und die Wiedergabe des Filmes angehalten.

```
on(release) {
    gotoAndStop(5);
}
```

gt (größer - zeichenfolgenspezifisch)

Syntax

`ausdruck1` `gt` *`ausdruck2`*

Argumente

`ausdruck1, ausdruck2` Zahlen, Zeichenfolgen oder Variablen.

Beschreibung

Operator (Vergleich); vergleicht *`ausdruck1`* mit *`ausdruck2`* und gibt `true` zurück, wenn *`ausdruck1`* größer als *`ausdruck2`* ist; andernfalls wird `false` zurückgegeben.

Player

Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen >-Operators empfohlen.

Siehe auch

„> (größer als)" auf Seite 215

_height

Syntax

`instanzname``._height`
`instanzname``._height =` *`wert`*`;`

Argumente

`instanzname` Der Instanzname einer Filmsequenz, für die die `_height`-Eigenschaft gesetzt oder abgerufen werden soll.

`wert` Eine Ganzzahl, die die Höhe des Filmes in Pixel angibt.

Beschreibung

Eigenschaft; setzt die Höhe des von den Inhalten eines Filmes ausgefüllten Raums und ruft diese ab. In früheren Versionen von Flash waren `_height` und `_width` schreibgeschützte Eigenschaften. In Flash 5 können diese Eigenschaften gesetzt werden.

Player

Flash 4 oder höher.

Beispiel

Das folgende Codebeispiel setzt Höhe und Breite einer Filmsequenz, wenn der Benutzer mit der Maus klickt.

```
onClipEvent(mouseDown) {
   _width=200;
   _height=200;
}
```

_highquality

Syntax

`_highquality = wert;`

Argumente

wert Der Grad des Anti-Alias, der dem Film zugewiesen wird. Geben Sie 2 (Optimal) an, um hohe Qualität mit ständig aktivierter Bitmapglättung zuzuweisen. Geben Sie 1 (hohe Qualität) an, um Anti-Alias zu aktivieren; hiermit werden Bitmaps geglättet, wenn der Film keine Animationen enthält. Geben Sie 0 (niedrige Qualität) an, um Anti-Alias zu deaktivieren.

Beschreibung

Eigenschaft (global); gibt den Grad des im aktuellen Film verwendeten Anti-Alias an.

Player

Flash 4 oder höher.

Siehe auch

„_quality“ auf Seite 344
„toggleHighQuality“ auf Seite 377

if

Syntax

`if(`*bedingung*`) {`

anweisung;

{

Argumente

bedingung Ein Ausdruck, der den Wert `true` oder `false` haben kann. Beispielsweise wertet `if(name == "Erica")`, die Variable `name` aus, um festzustellen, ob diese den Wert „Erica“ hat.

anweisungen Die Anweisungen, die ausgeführt werden sollen, wenn die Bedingung den Wert `true` hat.

Beschreibung

Aktion; wertet eine Bedingung aus, um die nächste Aktion in einem Film zu bestimmen. Wenn die Bedingung `true` ist, führt Flash die nachfolgenden Anweisungen aus. Verwenden Sie `if` für logische Verzweigungen in Skripten.

Player

Flash 4 oder höher.

Siehe auch

„else {“ auf Seite 263
„for“ auf Seite 266
„for..in“ auf Seite 267

ifFrameLoaded

Syntax

```
ifFrameLoaded(szene, bild) {
anweisung;}
```

```
ifFrameLoaded(bild) {
anweisung;}
```

Argumente

szene Die Szene, die abgefragt wird.

bild Die Bildnummer oder Bildbezeichnung, die vor Ausführung der nächsten Anweisung geladen werden soll.

Beschreibung

Aktion; überprüft, ob die Inhalte eines bestimmten Bilds lokal verfügbar sind. Verwenden Sie `ifFrameLoaded`, um eine einfache Animation wiederzugeben, während der restliche Film auf den lokalen Computer heruntergeladen wird. Der Unterschied zwischen der Verwendung von `_framesloaded` und `ifFrameLoaded` besteht darin, dass Sie bei `_framesloaded` `if`- oder `else`-Anweisungen hinzufügen können, wohingegen die `ifFrameLoaded`-Aktion die Angabe einer Anzahl von Bildern in einer einfachen Anweisung ermöglicht.

Player

Flash 3 oder höher. Die `ifFrameLoaded`-Aktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `_framesloaded`-Aktion empfohlen.

Siehe auch

„_framesloaded“ auf Seite 268

#include

Syntax

```
#include "dateiname.as";
```

Argumente

`dateiname.as` Der Name der einzuschließenden Datei; `.as` ist die empfohlene Dateierweiterung.

Beschreibung

Aktion; schließt den Inhalt der im Argument angegebenen Datei ein, wenn der Film getestet, veröffentlicht oder exportiert wird. Die `#include`-Aktion wird beim Testen, Veröffentlichen oder Exportieren aufgerufen. Die `#include`-Aktion wird bei einer Syntaxüberprüfung geprüft.

Player

Keine Angabe

Infinity

Syntax

```
Infinity
```

Argumente

Keine.

Beschreibung

Variable der obersten Ebene; eine vordefinierte Variable mit dem ECMA-262-Wert für Unendlichkeit.

Player

Flash 5 oder höher.

int

Syntax

```
int(wert);
```

Argumente

`wert` Eine Zahl, die zu einer Ganzzahl gerundet wird.

Beschreibung

Funktion; wandelt eine Dezimalzahl in den nächsten ganzzahligen Wert um.

Player

Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `Math.floor`-Methode empfohlen.

Siehe auch

„Math.floor“ auf Seite 300

isFinite

Syntax
`isFinite(ausdruck);`

Argumente
ausdruck Boolean, Variable oder ein anderer Ausdruck, der ausgewertet werden soll.

Beschreibung
Funktion der obersten Ebene; wertet das Argument aus und gibt `true` zurück, wenn es sich um eine endliche Zahl handelt. Die Funktion gibt `false` zurück, wenn es sich um unendlich oder negativ unendlich handelt. Ein Auftreten von unendlich oder negativ unendlich deutet auf mathematischen Fehlerzustand hin, beispielsweise eine Division durch 0.

Player
Flash 5 oder höher.

Beispiel
Es folgen Beispiele für Rückgabewerte von `isFinite`:

`isFinite(56)` gibt `true` zurück

`isFinite(Number.POSITIVE_INFINITY)` gibt `false` zurück

`isFinite(Number.POSITIVE_INFINITY)` gibt `false` zurück

isNaN

Syntax
`isNaN(ausdruck);`

Argumente
ausdruck Boolean, Variable oder ein anderer Ausdruck, der ausgewertet werden soll.

Beschreibung
Funktion der obersten Ebene; wertet das Argument aus und gibt `true` zurück, wenn der Wert keine Zahl ist (`NaN`). Dies deutet auf mathematische Fehler hin.

Player
Flash 5 oder höher.

Beispiel
Im folgenden wird der Rückgabewert von `isNan` demonstriert:

`isNan("Baum")` gibt `true` zurück

`isNan(56)` gibt `false` zurück

`isNaN(Number.POSITIVE_INFINITY)` gibt `false` zurück

Key (Objekt)

Das Key-Objekt ist ein Objekt der obersten Ebene, auf das ohne Verwendung eines Konstruktors zugegriffen werden kann. Mit den Methoden des Key-Objektes können Sie eine Benutzeroberfläche erstellen, die mit einer Standardtastatur gesteuert werden kann. Die Eigenschaften des Key-Objektes sind Konstanten, die die für Spielsteuerung am häufigsten verwendeten Tasten darstellen. In Anhang B „Tastatureingaben und Tastencodewerte“ finden Sie eine vollständige Liste der Tastencodewerte.

Beispiel

```
onClipEvent (enterFrame) {
    if(Key.isDown(Key.RIGHT)) {
        setProperty ("", _x, _x+10);
    }
}
or
onClipEvent (enterFrame) {
    if(Key.isDown(39)) {
        setProperty("", _x, _x+10);
    }
}
```

Zusammenfassung der Methoden für das Key-Objekt

Methode	Beschreibung
`getAscii;`	Gibt den ASCII-Wert der zuletzt gedrückten Taste zurück.
`getCode;`	Gibt den virtuellen Tastencode der zuletzt gedrückten Taste zurück.
`isDown;`	Gibt `true` zurück, wenn die im Argument angegebene Taste gedrückt ist.
`isToggled;`	Gibt `true` zurück, wenn die NUM- oder FESTSTELLTASTE aktiviert ist.

Zusammenfassung der Eigenschaften für das Key-Objekt

Alle Eigenschaften für das Key-Objekt sind Konstanten.

Eigenschaft	Beschreibung
`BACKSPACE`	Konstante, die dem Tastencodewert für die RÜCKSCHRITTTASTE (8) zugeordnet ist.
`CAPSLOCK`	Konstante, die dem Tastencodewert für die FESTSTELLTASTE (20) zugeordnet ist.

Eigenschaft	Beschreibung
`CONTROL`	Konstante, die dem Tastencodewert für STRG (17) zugeordnet ist.
`DELETEKEY`	Konstante, die dem Tastencodewert für ENTF (46) zugeordnet ist.
`DOWN`	Konstante, die dem Tastencodewert für die NACH-UNTEN-TASTE (40) zugeordnet ist.
`END`	Konstante, die dem Tastencodewert für ENDE (35) zugeordnet ist.
`ENTER`	Konstante, die dem Tastencodewert für die EINGABETASTE (13) zugeordnet ist.
`ESCAPE`	Konstante, die dem Tastencodewert für ESC (27) zugeordnet ist.
`HOME`	Konstante, die dem Tastencodewert für POS1 (36) zugeordnet ist.
`INSERT`	Konstante, die dem Tastencodewert für EINFG (45) zugeordnet ist.
`LEFT`	Konstante, die dem Tastencodewert für die NACH-LINKS-TASTE (37) zugeordnet ist.
`PGDN`	Konstante, die dem Tastencodewert für die BILD-AB-TASTE (34) zugeordnet ist.
`PGUP`	Konstante, die dem Tastencodewert für die BILD-AUF-TASTE (33) zugeordnet ist.
`RIGHT`	Konstante, die dem Tastencodewert für die NACH-RECHTS-TASTE (39) zugeordnet ist.
`SHIFT`	Konstante, die dem Tastencodewert für die Umschalttaste (16) zugeordnet ist.
`SPACE`	Konstante, die dem Tastencodewert für die LEERTASTE (32) zugeordnet ist.
`TAB`	Konstante, die dem Tastencodewert für die TAB-TASTE (9) zugeordnet ist.
`UP`	Konstante, die dem Tastencodewert für die NACH-OBEN-TASTE (38) zugeordnet ist.

Key.BACKSPACE

Syntax

```
Key.BACKSPACE
```

Argumente

Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die RÜCKSCHRITTTASTE (8) zugeordnet ist.

Player
Flash 5 oder höher.

Key.CAPSLOCK

Syntax
`Key.CAPSLOCK`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die FESTSTELLTASTE (20) zugeordnet ist.

Player
Flash 5 oder höher.

Key.CONTROL

Syntax
`Key.CONTROL`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für STRG (17) zugeordnet ist.

Player
Flash 5 oder höher.

Key.DELETEKEY

Syntax
`Key.DELETEKEY`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für ENTF (46) zugeordnet ist.

Player
Flash 5 oder höher.

Key.DOWN

Syntax
`Key.DOWN`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die NACH-UNTEN-TASTE (40) zugeordnet ist.

Player
Flash 5 oder höher.

Key.END

Syntax
`Key.END`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für ENDE (35) zugeordnet ist.

Player
Flash 5 oder höher.

Key.ENTER

Syntax
`Key.ENTER`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die EINGABETASTE (13) zugeordnet ist.

Player
Flash 5 oder höher.

Key.ESCAPE

Syntax
```
Key.ESCAPE
```

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für ESC (27) zugeordnet ist.

Player
Flash 5 oder höher.

Key.getAscii

Syntax
```
Key.getAscii();
```

Argumente
Keine.

Beschreibung
Methode; gibt den ASCII-Code der zuletzt gedrückten bzw. losgelassenen Taste zurück.

Player
Flash 5 oder höher.

Key.getCode

Syntax
```
Key.getCode();
```

Argumente
Keine.

Beschreibung
Methode; gibt den Tastencodewert der zuletzt gedrückten Taste zurück. Mit Hilfe der Informationen in Anhang B „Tastatureingaben und Tastencodewerte" können Sie den zurückgegebenen Tastencodewert der virtuellen Taste einer Standardtastatur zuordnen.

Player
Flash 5 oder höher.

Key.HOME

Syntax
`Key.HOME`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für POS1 (36) zugeordnet ist.

Player
Flash 5 oder höher.

Key.INSERT

Syntax
`Key.INSERT`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für EINFG (45) zugeordnet ist.

Player
Flash 5 oder höher.

Key.isDown

Syntax
`Key.isDown(`*`tastencode`*`);`

Argumente
`tastencode` Der einer bestimmten Taste zugeordnete Tastencodewert oder eine Eigenschaft des Key-Objektes, die einer bestimmten Taste zugeordnet ist. Anhang B „Tastatureingaben und Tastencodewerte" enthält alle Tastencodes, die Tasten auf einer Standardtastatur zugeordnet sind.

Beschreibung
Methode; gibt `true` zurück, wenn die in *`tastencode`* angegebene Taste gedrückt ist. Auf Macintosh-Computern sind die Tastencodewerte für die FESTSTELL-TASTE und die NUM-TASTE identisch.

Player
Flash 5 oder höher.

Key.isToggled

Syntax
`Key.isToggled(`*`tastencode`*`)`

Argumente
`tastencode` Der Tastencode für die FESTSTELLTASTE (20) oder die NUM-TASTE (144).

Beschreibung
Methode; gibt `true` zurück, wenn die FESTSTELLTASTE oder die NUM-TASTE aktiviert (umgeschaltet) wird. Auf Macintosh-Computern sind die Tastencodewerte für diese Tasten identisch.

Player
Flash 5 oder höher.

Key.LEFT

Syntax
`Key.LEFT`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die NACH-LINKS-TASTE (37) zugeordnet ist.

Player
Flash 5 oder höher.

Key.PGDN

Syntax
`Key.PGDN`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die BILD-AB-TASTE (34) zugeordnet ist.

Player
Flash 5 oder höher.

Key.PGUP

Syntax
`Key.PGUP`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die BILD-AUF-TASTE (33) zugeordnet ist.

Player
Flash 5 oder höher.

Key.RIGHT

Syntax
`Key.RIGHT`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die NACH-RECHTS-TASTE (39) zugeordnet ist.

Player
Flash 5 oder höher.

Key.SHIFT

Syntax
`Key.SHIFT`

Argumente
Keine.

Beschreibung
Eigenschaft; Konstante, die dem Tastencodewert für die Umschalttaste (16) zugeordnet ist.

Player
Flash 5 oder höher.

Key.SPACE

Syntax

`Key.SPACE`

Argumente

Keine.

Beschreibung

Eigenschaft; Konstante, die dem Tastencodewert für die LEERTASTE (32) zugeordnet ist.

Player

Flash 5 oder höher.

Key.TAB

Syntax

`Key.TAB`

Argumente

Keine.

Beschreibung

Eigenschaft; Konstante, die dem Tastencodewert für die TAB-TASTE (9) zugeordnet ist.

Player

Flash 5 oder höher.

Key.UP

Syntax

`Key.UP`

Argumente

Keine.

Beschreibung

Eigenschaft; Konstante, die dem Tastencodewert für die NACH-OBEN-TASTE (38) zugeordnet ist.

Player

Flash 5 oder höher.

le (kleiner oder gleich - zeichenfolgenspezifisch)

Syntax

```
ausdruck1 le ausdruck2
```

Argumente

ausdruck1,ausdruck2 Zahlen, Zeichenfolgen oder Variablen.

Beschreibung

Operator (Vergleich); vergleicht *ausdruck1* mit *ausdruck2* und gibt `true` zurück, wenn *ausdruck1* kleiner oder gleich *ausdruck2* ist; andernfalls wird `false` zurückgegeben.

Player

Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen `<=`-Operators empfohlen.

Siehe auch

„<= (kleiner oder gleich)“ auf Seite 212

length

Syntax

```
length(ausdruck);
```

```
length(variable);
```

Argumente

ausdruck Beliebige Zeichenfolge.

variable Der Name einer Variablen.

Beschreibung

Zeichenfolgenfunktion; gibt die Länge der angegebenen Zeichenfolge oder des Variablennamens zurück.

Player

Flash 4 oder höher. Diese Funktion gilt wie alle Zeichenfolgenfunktionen in Flash 5 als veraltet. Stattdessen wird die Verwendung der Methoden und der `length`-Eigenschaft des String-Objektes zur Ausführung der entsprechenden Operationen empfohlen.

Beispiel

Im folgenden Beispiel wird der Wert der Zeichenfolge `Hallo` zurückgegeben.

```
length("Hallo")
```

Das Ergebnis ist 5.

Siehe auch

„" " (Zeichenfolgentrennzeichen)“ auf Seite 365
„String.length“ auf Seite 370

_level

Syntax

`_levelN;`

Argumente

`N` Eine nicht-negative Ganzahl zur Angabe der Tiefenebene. Die Standardeinstellung für `_level` lautet 0, d. h., der Film, der sich am unteren Ende der Hierarchie befindet.

Beschreibung

Eigenschaft; eine Referenz auf die Stammfilmzeitleiste der Ebene *`levelN`*. Filme müssen mit Hilfe der `loadMovie`-Aktion geladen werden, bevor Sie diese mit der `_level`-Eigenschaft ansprechen können.

Im Flash Player werden Filme mit einer Nummer versehen, und zwar in der Reihenfolge, in der sie geladen werden. Der zuerst geladene Film wird in die unterste Ebene, also in die Ebene 0 geladen. Mit dem Film in der Ebene 0 wird die Bildrate, die Hintergrundfarbe und das Bildformat aller im Folgenden geladenen Filme festgelegt. Filme werden danach in Ebenen mit höherer Nummer über dem Film in Ebene 0 gestapelt. Die Ebene, auf der sich eine Filmsequenz befindet, wird auch als Tiefenebene oder Tiefe bezeichnet.

Player

Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird die Zeitleiste des Filmes in Ebene 0 angehalten.

```
_level0.stop();
```

Im folgenden Beispiel wird die Zeitleiste des Filmes in Ebene 4 an Bild 5 gesendet. Der Film in Ebene 4 muss vorher mit der `loadMovie`-Aktion geladen werden.

```
_level4.gotoAndStop(5);
```

Siehe auch

„loadMovie“ auf Seite 291
„MovieClip.swapDepths“ auf Seite 322

loadMovie

Syntax

`loadMovie(url [,position/ziel, variablen]]);`

Argumente

`url` Ein absoluter oder relativer URL für die zu ladende SWF-Datei. Ein relativer Pfad muss relativ zur SWF-Datei angegeben werden. Der URL muss sich in der gleichen Subdomäne befinden wie der URL, unter dem der Film momentan abgelegt ist. Um SWF-Dateien in Flash Player verwenden oder in der Flash Erstellungsumgebung im Filmtestmodus testen zu können, müssen diese im gleichen Ordner gespeichert sein. Die Dateinamen dürfen außerdem keine Laufwerks- und Ordnerbezeichnungen enthalten.

`ziel` Ein optionales Argument, mit dem eine Zielfilmsequenz angegeben wird, die durch den geladenen Film ersetzt wird. Der geladene Film übernimmt die Eigenschaften für Position, Drehung und Skalierung der Filmsequenz, die als Ziel angegeben wird. Wenn Sie `ziel` angeben, entspricht dies dem Angeben der `position` (Ebene) des ZielFilmes. Es empfiehlt sich daher nicht, beides festzulegen.

`position` Ein optionales Argument, mit dem die Ebene angegeben wird, in die der Film geladen wird. Der geladene Film übernimmt die Eigenschaften für Position, Drehung und Skalierung der Filmsequenz, die als Ziel angegeben wird. Wenn Sie den neuen Film zusätzlich zu vorhandenen Filmen laden möchten, legen Sie eine Ebene fest, die nicht von einem anderen Film belegt ist. Wenn Sie einen vorhandenen Film durch den geladenen Film ersetzen möchten, legen Sie eine Ebene fest, die momentan durch einen anderen Film belegt ist. Zum Ersetzen des ursprünglichen Filmes und zum Entladen aller Ebenen laden Sie den neuen Film in die Ebene 0. Der Film in der Ebene 0 legt die Bildrate, die Hintergrundfarbe sowie das Bildformat für alle anderen geladenen Filme fest.

`variablen` Ein optionales Argument, das eine Methode für das Senden von Variablen angibt, die dem zu ladenden Film zugeordnet sind. Bei diesem Argument muss es sich um die Zeichenfolge „GET“ oder „POST“ handeln. Wenn es keine Variablen gibt, lassen Sie dieses Argument aus; andernfalls geben Sie mit Hilfe der `GET`- oder `POST`-Methode an, ob Variablen geladen werden sollen. Bei `GET` werden die Variablen am Ende des URL angehängt. `GET` wird bei einer geringen Anzahl von Variablen verwendet. Bei `POST` werden die Variablen in einer separaten HTTP-Kopfzeile gesendet. `POST` wird bei langen Zeichenfolgen für Variablen verwendet.

Beschreibung

Aktion; gibt zusätzliche Filme wieder, ohne Flash Player zu schließen. Normalerweise zeigt der Flash Player nur einen einzelnen Flash Player-Film (SWF-Datei) an und wird dann geschlossen. Mit der `loadMovie`-Aktion können Sie mehrere Filme auf einmal abspielen bzw. zwischen einzelnen Filmen hin- und herschalten, ohne ein weiteres HTML-Dokument laden zu müssen.

Sie können auch Filme in Ebenen laden, in die bereits SWF-Dateien geladen wurden. In diesem Fall wird die bestehende SWF-Datei durch den neuen Film ersetzt. Wenn Sie einen neuen Film in Ebene 0 laden, werden alle anderen Ebenen entladen, und Ebene 0 wird durch die neue Datei ersetzt. Behalten Sie mit Hilfe der `loadVariables`-Aktion den aktiven Film bei, und aktualisieren Sie die Variablen mit neuen Werten.

Mit der `unloadMovie`-Aktion können Sie Filme entfernen, die mit der `loadMovie`-Aktion geladen wurden.

Player
Flash 3 oder höher.

Beispiel
Diese `loadMovie`-Anweisung ist an die Navigationsschaltfläche mit der Beschriftung „Produkte" gebunden. Es gibt eine unsichtbare Filmsequenz auf der Bühne mit dem Instanznamen `dropZone`. Die loadMovie-Aktion verwendet diese Filmsequenz als Zielparameter zum Laden der Produkte in der SWF-Datei in die korrekte Position auf der Bühne.

```
on(release) {
    loadMovie("produkte.swf",_root.dropZone);
}
```

Siehe auch
„unloadMovie" auf Seite 380
„_level" auf Seite 290

loadVariables

Syntax
`loadVariables (`*`url`*` ,`*`position`*`[, `*`variablen`*`]);`

Argumente
url Ein absoluter oder relativer URL, an dem sich die Variablen befinden. Der Host für den URL muss sich in derselben Subdomäne wie der Film befinden, wenn der Zugriff von einem Webbrowser erfolgt.

position Eine Ebene oder ein Ziel für die Variablen. Im Flash Player werden Filmdateien mit einer Nummer versehen, und zwar in der Reihenfolge, in der sie geladen werden. Der erste Film wird in die unterste Ebene (Ebene 0) geladen. In der `loadMovie`-Aktion müssen Sie eine Ebenennummer für die einzelnen aufeinanderfolgenden Filme angeben. Dieses Argument ist optional.

variablen Ein optionales Argument, das eine Methode für das Senden von Variablen angibt. Wenn es keine Variablen gibt, lassen Sie dieses Argument aus; andernfalls geben Sie mit Hilfe der `GET`- oder `POST`-Methode an, ob Variablen geladen werden sollen. Bei `GET` werden die Variablen am Ende des URL angehängt. `GET` wird bei einer geringen Anzahl von Variablen verwendet. Bei `POST` werden die Variablen in einer separaten HTTP-Kopfzeile gesendet. `POST` wird bei langen Zeichenfolgen für Variablen verwendet.

Beschreibung

Aktion; liest Daten aus einer externen Datei, beispielsweise aus einer Textdatei oder Text, der von einem CGI-Skript, mit Active Server Pages (ASP) oder Personal Home Page (PHP) generiert wurde, und setzt die Werte für Variablen in einer Filmsequenz oder einem Film. Mit dieser Aktion können Variablen im aktiven Film auch mit neuen Werten aktualisiert werden.

Der Text unter dem angegebenen URL muss im Standard-MIME-Format *application/x-www-urlformencoded* (einem Standardformat, das von CGI-Skripts verwendet wird) vorliegen. Der Film und die Variablen, die geladen werden sollen, müssen sich in derselben Subdomäne befinden. Es kann eine beliebige Anzahl an Variablen angegeben werden. Mit dem folgenden Ausdruck werden z. B. mehrere Variablen definiert:

```
company=Macromedia&address=600+Townsend&city=San+Francisco&zip=94
103
```

Player

Flash 4 oder höher.

Beispiel

In diesem Beispiel werden Informationen aus einer Textdatei in die Textfelder auf der Hauptzeitleiste (Ebene 0) geladen. Die Variablennamen der Textfelder müssen mit den Variablennamen in der Datei data.txt übereinstimmen.

```
on(release) {
    loadVariables("data.txt", 0);
}
```

Siehe auch

„getURL“ auf Seite 272
„MovieClip.loadMovie“ auf Seite 317
„MovieClip.loadVariables“ auf Seite 318

lt (kleiner als - zeichenfolgenspezifisch)

Syntax

ausdruck1 `lt` *ausdruck2*

Argumente

ausdruck1, ausdruck2 Zahlen, Zeichenfolgen oder Variablen.

Beschreibung
Operator (Vergleich); vergleicht *ausdruck1* mit *ausdruck2* und gibt `true` zurück, wenn *ausdruck1* kleiner als *ausdruck2* ist; andernfalls wird `false` zurückgegeben.

Player
Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen <-Operators (kleiner als) empfohlen.

Siehe auch
„< (kleiner als)“ auf Seite 210

Math (Objekt)

Das Math-Objekt ist ein Objekt der obersten Ebene, auf das ohne Verwendung eines Konstruktors zugegriffen werden kann.

Verwenden Sie die Methoden und Eigenschaften dieses Objektes, um auf mathematische Konstanten und Funktionen zuzugreifen und diese zu bearbeiten. Alle Eigenschaften und Methoden des Math-Objektes sind statisch und müssen mit der Syntax `Math.methode(`*argument*`)` oder `Math.konstante` aufgerufen werden. In ActionScript sind Konstanten mit maximaler Genauigkeit der doppeltgenauen IEEE-754-Gleitkommazahlen definiert.

Das Math-Objekt wird vom Flash 5 Player vollständig unterstützt. Im Flash 4 Player funktionieren die Methoden des Math-Objektes, sie werden jedoch mit Annäherungen emuliert und sind deshalb nicht so genau wie die nicht emulierten Math-Funktionen, die vom Flash 5 Player unterstützt werden.

Einige der Methoden des Math-Objektes akzeptieren die Radianten eines Winkels als Argument. Sie können mit der untenstehenden Gleichung die Werte der Radianten berechnen oder einfach die Gleichung (durch Eingeben eines Werts für Gradzahlen) anstelle des Arguments für die Radianten übergeben.

Zum Berechnen eines Radiantwerts verwenden Sie die folgende Formel:

```
bogenlaenge = Math.PI/180 * gradzahl
```

Im folgenden Beispiel wird die Gleichung als Argument zum Berechnen des Sinus eines Winkels von 45 Grad übergeben:

`Math.SIN(Math.PI/180 * 45)` ist identisch mit `Math.SIN(.7854)`

Zusammenfassung der Methoden für das Math-Objekt

Methode	Beschreibung
`abs`	Berechnet einen absoluten Wert.
`acos`	Berechnet einen Arkuskosinus.
`asin`	Berechnet einen Arkussinus.
`atan`	Berechnet einen Arkustangens.
`atan2`	Berechnet einen Winkel von der x-Achse bis zum Punkt.
`ceil`	Rundet eine Zahl bis zur nächsten Ganzzahl auf.
`cos`	Berechnet einen Kosinus.
`exp`	Berechnet einen Exponentialwert.
`floor`	Rundet eine Zahl auf die nächste Ganzzahl ab.
`log`	Berechnet einen natürlichen Logarithmus.
`max`	Gibt die größere der zwei Ganzzahlen zurück.
`min`	Gibt die kleinere der zwei Ganzzahlen zurück.
`pow`	Berechnet die Potenz *x* hoch *y*.
`random`	Gibt eine Pseudozufallszahl zwischen 0,0 und 1,0 zurück.
`round`	Rundet auf die nächste Ganzzahl.
`sin`	Berechnet einen Sinus.
`sqrt`	Berechnet eine Quadratwurzel.
`tan`	Berechnet einen Tangens.

Zusammenfassung der Eigenschaften für das Math-Objekt

Alle Eigenschaften für das Math-Objekt sind Konstanten.

Eigenschaft	Beschreibung
`E`	Eulersche Konstante und die Basis natürlicher Logarithmen (ungefähr 2,718).
`LN2`	Der natürliche Logarithmus von 2 (ungefähr 0,693).
`LOG2E`	Der Logarithmus von *e* zur Basis 2 (ungefähr 1,442).
`LN10`	Der natürliche Logarithmus von 10 (ungefähr 2,302).

Eigenschaft	Beschreibung
`LOG10E`	Der Logarithmus von *e* zur Basis 10 (ungefähr 0,434).
`PI`	Das Verhältnis von Kreisumfang zu -durchmesser (ungefähr 3,14159).
`SQRT1_2`	Der Kehrwert der Quadratwurzel von 1/2 (ungefähr 0,707).
`SQRT2`	Die Quadratwurzel von 2 (ungefähr 1,414).

Math.abs

Syntax

```
Math.abs(x);
```

Argumente

x Beliebige Zahl.

Beschreibung

Methode; berechnet einen absoluten Wert für die im Argument *x* angegebene Zahl und gibt diesen zurück.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.acos

Syntax

```
Math.acos(x);
```

Argumente

x Eine Zahl zwischen -1.0 und 1.0.

Beschreibung

Methode; berechnet den Arkuskosinus für die im Argument *x* angegebene Zahl in Radianten und gibt diesen zurück.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.asin

Syntax

```
Math.asin(x);
```

Argumente

x Eine Zahl zwischen -1,0 und 1,0.

Beschreibung

Methode; berechnet den Arkussinus für die im Argument *x* angegebene Zahl in Radianten und gibt diesen zurück.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.atan

Syntax

```
Math.atan(x);
```

Argumente

x Beliebige Zahl.

Beschreibung

Methode; berechnet den Arkustangens für die im Argument *x* angegebene Zahl und gibt diesen zurück. Der Ausgabewert liegt zwischen negativ Pi dividiert durch 2 und positiv Pi dividiert durch 2.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.atan2

Syntax

```
Math.atan2(y, x);
```

Argumente

x Eine Zahl zur Angabe der *x*-Koordinate des Punkts.

y Eine Zahl zur Angabe der *y*-Koordinate des Punkts.

Beschreibung

Methode; berechnet den Arkustanges von *y*/*x* in Radianten und gibt diesen zurück. Der Rückgabewert stellt den Winkel gegenüber dem rechten Winkel in einem rechtwinkligen Dreieck dar, wobei *x* die anliegende Seitenlänge und *y* die gegenüberliegende Seitenlänge ist.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.ceil

Syntax

```
Math.ceil(x);
```

Argumente

x Eine Zahl oder ein Ausdruck.

Beschreibung

Methode; gibt die Obergrenze der angegebenen Zahl oder des Ausdrucks zurück. Die Obergrenze einer Zahl ist die nächstliegende Ganzzahl, die größer oder gleich der Zahl ist.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.cos

Syntax

```
Math.cos(x);
```

Argumente

x Ein Winkel gemessen in Radianten.

Beschreibung

Methode; gibt den Kosinus (einen Wert von -1,0 bis 1,0) des im Argument *x* angegebenen Winkels zurück. Der Winkel *x* muss in Radianten angegeben werden. Informationen zum Berechnen von Radianten finden Sie in der Einführung zum Math-Objekt.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Beispiel

Math.E

Syntax

```
Math.E
```

Argumente

Keine.

Beschreibung

Konstante; eine mathematische Konstante für die Basis des natürlichen Logarithmus als *e* ausgedrückt. Der ungefähre Wert von *e* ist 2,71828.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.exp

Syntax

```
Math.exp(x);
```

Argumente

`x` Der Exponent; eine Zahl oder ein Ausdruck.

Beschreibung

Methode; gibt den Wert der Basis des natürlichen Logarithmus (*e*) mit der Potenz des im Argument `x` angegebenen Exponenten zurück. Die Konstante `Math.E` kann den Wert *e* zur Verfügung stellen.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.floor

Syntax

```
Math.floor(x);
```

Argumente

x Eine Zahl oder ein Ausdruck.

Beschreibung

Methode; gibt die Untergrenze der angegebenen Zahl oder des Ausdrucks zurück. Die Untergrenze einer Zahl ist die nächstliegende Ganzzahl, die kleiner oder gleich der angegebenen Zahl oder dem Ausdruck ist.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Beispiel

Im folgenden Beispiel wird der Wert 12 zurückgegeben.

```
Math.floor(12.5);
```

Math.log

Syntax

```
Math.log(x);
```

Argumente

x Eine Zahl oder ein Ausdruck mit einem Wert größer als 0.

Beschreibung

Methode; gibt den natürlichen Logarithmus des Arguments *x* zurück.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.LOG2E

Syntax

```
Math.LOG2E
```

Argumente

Keine.

Beschreibung

Konstante; eine mathematische Konstante für den Logarithmus zur Basis 2 der Konstante *e* (`Math.E`), der als $\log_e 2$ ausgedrückt wird und den ungefähren Wert 1,442695040888963387 hat.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.LOG10E

Syntax

```
Math.LOG10E
```

Argumente

Keine.

Beschreibung

Konstante; eine mathematische Konstante für den Logarithmus zur Basis 10 der Konstante *e* (`Math.E`), der als $\log_{10} e$ ausgedrückt wird und den ungefähren Wert 0,43429448190325181667 hat.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.LN2

Syntax

```
Math.LN2
```

Argumente

Keine.

Beschreibung

Konstante; eine mathematische Konstante für den natürlichen Logarithmus von 2, der als $\log_e 2$ ausgedrückt wird und den ungefähren Wert 0,69314718055994528623 hat.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.LN10

Syntax
`Math.LN10`

Argumente
Keine.

Beschreibung
Konstante; eine mathematische Konstante für den natürlichen Logarithmus von 10, der als $\log_e 10$ ausgedrückt wird und den ungefähren Wert 2,3025850929940459011 hat.

Player
Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.max

Syntax
`Math.max(x , y);`

Argumente
x Eine Zahl oder ein Ausdruck.

y Eine Zahl oder ein Ausdruck.

Beschreibung
Methode; wertet *x* und *y* aus und gibt den größeren Wert zurück.

Player
Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.min

Syntax
`Math.min(x , y);`

Argumente
x Eine Zahl oder ein Ausdruck.

y Eine Zahl oder ein Ausdruck.

Beschreibung

Methode; wertet *x* und *y* aus und gibt den kleineren Wert zurück.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.PI

Syntax

```
Math.PI
```

Argumente

Keine.

Beschreibung

Konstante; eine mathematische Konstante für das Verhältnis des Umfangs eines Kreises zu seinem Durchmesser der als Pi mit dem Wert 3,14159265358979 ausgedrückt wird.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.pow

Syntax

```
Math.pow(x , y);
```

Argumente

x   Die Basis einer Potenz.

y   Eine Zahl, die den Exponenten für die Basis *x* einer Potenz angibt.

Beschreibung

Methode; berechnet *x* hoch *y* (x^y) *und gibt das Ergebnis zurück.*

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.random

Syntax
`Math.random();`

Argumente
Keine.

Beschreibung
Methode; gibt eine Pseudozufallszahl zwischen 0,0 und 1,0 zurück.

Player
Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Siehe auch
„random“ auf Seite 345

Math.round

Syntax
`Math.round(x);`

Argumente
x Beliebige Zahl.

Beschreibung
Methode; rundet den Wert des Arguments *x* zur nächstliegenden Ganzzahl auf oder ab und gibt diese zurück.

Player
Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.sin

Syntax
`Math.sin(x);`

Argumente
x Ein Winkel gemessen in Radianten.

Beschreibung

Methode; berechnet den Sinus des angegebenen Winkels in Radianten und gibt den Wert zurück. Informationen zum Berechnen von Radianten finden Sie in der Einführung zum Math-Objekt.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Siehe auch

„`Math (Objekt)`“ auf Seite 294

Math.sqrt

Syntax

```
Math.sqrt(x);
```

Argumente

x Eine beliebige Zahl oder ein Ausdruck größer oder gleich 0.

Beschreibung

Methode; berechnet die Quadratwurzel der angegebenen Zahl und gibt diese zurück.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.SQRT1_2

Syntax

```
Math.SQRT1_2
```

Argumente

Keine.

Beschreibung

Konstante; eine mathematische Konstante für den Kehrwert der Quadratwurzel von einhalb (1/2) mit dem ungefähren Wert 0,707106781186.

Player

Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.SQRT2

Syntax
`Math.SQRT2`

Argumente
Keine.

Beschreibung
Konstante; eine mathematische Konstante für die Quadratwurzel von 2, die als mit dem ungefähren Wert 1,414213562373 ausgedrückt wird.

Player
Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

Math.tan

Syntax
`Math.tan(x);`

Argumente
x Ein Winkel gemessen in Radianten.

Beschreibung
Methode; berechnet den Tangens des angegebenen Winkels und gibt diesen zurück. Informationen zum Berechnen von Radianten finden Sie in der Einführung zum Math-Objekt.

Player
Flash 5 oder höher. Im Flash 4 Player werden die Methoden und Eigenschaften des Math-Objektes mit Annäherungen emuliert und sind deshalb möglicherweise nicht so genau wie die nicht emulierten mathematischen Funktionen des Flash 5 Player.

maxscroll

Syntax
`variablenname.maxscroll = x`

Argumente
variablenname Der Name einer Variable, die an ein Textfeld gebunden ist.

x Die Zeilennummer, die den größten möglichen Wert für die `scroll`-Eigenschaft entsprechend der Höhe des Textfeldes darstellt. Dies ist ein schreibgeschützter Wert, der durch Flash gesetzt wird.

Beschreibung
Eigenschaft; eine schreibgeschützte Eigenschaft, die `scroll`-Eigenschaft verwendet, um die Anzeige von Informationen in einem Textfeld zu steuern. Diese Eigenschaft kann abgerufen, aber nicht geändert werden.

Player
Flash 4 oder höher.

Siehe auch
„`scroll`“ `auf Seite 348`

mbchr

Syntax
```
mbchr(zahl);
```

Argumente
`zahl` Die Zahl, die in ein Multibyte-Zeichen umgewandelt werden soll.

Beschreibung
Zeichenfolgenfunktion; wandelt eine ASCII-Codenummer in ein Multibyte-Zeichen um.

Player
Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `String.fromCharCode`-Methode empfohlen.

Siehe auch
„`String.fromCharCode`“ `auf Seite 369`

mblength

Syntax
```
mblength(zeichenfolge);
```

Argumente
`zeichenfolge` Eine Zeichenfolge.

Beschreibung
Zeichenfolgenfunktion; gibt die Länge der Multibyte-Zeichenfolge zurück.

Player
Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung von String-Objekt und -Methoden empfohlen.

mbord

Syntax
`mbord(zeichen);`

Argumente
zeichen Das in eine Multibyte-Zahl umzuwandelnde Zeichen.

Beschreibung
Zeichenfolgenfunktion; wandelt das angegebene Zeichen in eine Multibyte-Zahl um.

Player
Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `String.charCodeAt`-Methode empfohlen.

Siehe auch
„String.fromCharCode" auf Seite 369

mbsubstring

Syntax
`mbsubstring(wert, index, anzahl);`

Argumente
wert Die Multibyte-Zeichenfolge, aus der eine neue Multibyte-Zeichenfolge extrahiert werden soll.

index Die Nummer des ersten zu extrahierenden Zeichens.

anzahl Die Anzahl der Zeichen ohne das Indexzeichen, die die extrahierte Zeichenfolge umfassen soll.

Beschreibung
Zeichenfolgenfunktion; extrahiert eine neue Multibyte-Zeichenfolge aus einer Multibyte-Zeichenfolge.

Player
Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `string.substr`-Methode empfohlen.

Siehe auch
„String.substr" auf Seite 372

Mouse (Objekt)

Verwenden Sie die Methoden des Mouse-Objektes, um den Mauszeiger im Film ein- oder auszublenden. Der Mauszeiger wird gemäß Standardeinstellung eingeblendet, Sie können diesen aber ausblenden und einen benutzerdefinierten Mauszeiger einsetzen, den Sie mit einer Filmsequenz erstellen.

Zusammenfassung der Mouse-Methode

Methode	Beschreibung
`hide`	Blendet den Mauszeiger im Film aus.
`show`	Zeigt den Mauszeiger im Film an.

Mouse.hide

Syntax

`Mouse.hide();`

Argumente

Keine.

Beschreibung

Methode; blendet den Mauszeiger in einem Film aus. Der Mauszeiger wird gemäß Standardeinstellung angezeigt.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Code, der einer Filmsequenz in der Hauptzeitleiste zugeordnet ist, werden der Standardmauszeiger ausgeblendet und die Positionen *x* und *y* der Filmsequenzinstanz `customCursor` auf die Mauspositionen *x* und *y* in der Hauptzeitleiste gesetzt:

```
onClipEvent(enterFrame){
   Mouse.hide();
   customCursorMC_x = _root._xmouse;
   customCursorMC_y = _root._ymouse;
}
```

Siehe auch

„`_xmouse`“ auf Seite 413
„`_ymouse`“ auf Seite 414
„`Mouse.show`“ auf Seite 309

Mouse.show

Syntax

`Mouse.show();`

Argumente

Keine.

Beschreibung
Methode; blendet den Mauszeiger in einem Film ein. Der Mauszeiger wird gemäß Standardeinstellung angezeigt.

Player
Flash 5 oder höher.

Siehe auch
„_xmouse“ auf Seite 413
„_ymouse“ auf Seite 414
„Mouse.show“ auf Seite 309

MovieClip (Objekt)

Die Methoden für das MovieClip-Objekt bieten dieselben Funktionen wie die Standardaktionen für Filmsequenzen. Es gibt außerdem einige zusätzliche Methoden, die Funktionen bieten, die bei den Standardaktionen im Bedienfeld „Aktionen“ in der Kategorie „Aktionen“ nicht zur Verfügung stehen. Sie müssen keine Konstruktormethode verwenden, um die Methoden des MovieClip-Objektes aufzurufen. Stattdessen verweisen Sie auf Instanzen von Filmsequenzen über den Namen und mit der folgenden Syntax:

```
eineFilmsequenz.play();
eineFilmsequenz.gotoAndPlay(3);
```

Zusammenfassung der Methoden für das MovieClip-Objekt

Methode	Beschreibung
attachMovie	Fügt einen Film der Bibliothek hinzu.
duplicateMovieClip	Dupliziert die angegebene Filmsequenz.
getBounds	Gibt die jeweils kleinsten und größten Werte für die Koordinaten *x* und *y* eines Filmes in einem bestimmten Koordinatensystem zurück.
getBytesLoaded	Gibt die Anzahl der geladenen Bytes für die angegebene Filmsequenz zurück.
getBytesTotal	Gibt die Größe der Filmsequenz in Byte zurück.
getURL	Ruft ein Dokument von einer URL ab.
globalToLocal	Wandelt das Punktobjekt von den Bühnenkoordinaten in die lokalen Koordinaten der angegebenen Filmsequenz um.
gotoAndPlay	Sendet den Abspielkopf zum angegebenen Bild in der Filmsequenz und gibt den Film wieder.
gotoAndStop	Sendet den Abspielkopf zum angegebenen Bild in der Filmsequenz und hält den Film an.

Methode	Beschreibung
`hitTest`	Gibt `true` zurück, wenn der Begrenzungsrahmen der angegebenen Filmsequenz den Begrenzungsrahmen der Zielfilmsequenz überschneidet.
`loadMovie`	Lädt den angegebenen Film in die Filmsequenz.
`loadVariables`	Lädt Variablen von einem URL oder einem anderen Speicherort in die Filmsequenz.
`localToGlobal`	Wandelt ein Punktobjekt aus den lokalen Koordinaten der Filmsequenz in die globalen Bühnenkoordinaten um.
`nextFrame`	Sendet den Abspielkopf zum nächsten Bild der Filmsequenz.
`play`	Gibt die angegebene Filmsequenz wieder.
`prevFrame`	Sendet den Abspielkopf zum vorigen Bild der Filmsequenz.
`removeMovieclip`	Entfernt die Filmsequenz aus der Zeitleiste, sofern jene mit einer duplicateMovieClip-Aktion oder -Methode oder der attachMovie-Methode erstellt wurde.
`startDrag`	Gibt an, dass eine Filmsequenz mit der Maus gezogen werden kann, und beginnt die Filmsequenz zu ziehen.
`stop`	Stoppt die Wiedergabe des aktuellen Filmes.
`stopDrag`	Stoppt den Ziehvorgang aller Filmsequenzen, die momentan gezogen werden.
`swapDepths`	Vertauscht die Tiefenebene des angegebenen Filmes mit dem Film auf der angegebenen Tiefenebene.
`unloadMovie`	Entfernt einen mit `loadMovie` geladenen Film.

MovieClip.attachMovie

Syntax

eineFilmsequenz`.attachMovie(`*idName*`, `*neuerName*`, `*tiefe*`);`

Argumente

idName Der Name des Filmes in der Bibliothek, der angehängt werden soll. Dieser Name wird im Dialogfeld „Eigenschaften Symbolverknüpfung“ im Feld „Bezeichner“ eingegeben.

neuerName Ein eindeutiger Instanzname für die angehängte Filmsequenz.

tiefe Eine Ganzzahl zur Angabe der Tiefenebene, in der der Film abgelegt wird.

Beschreibung

Methode; erstellt eine neue Instanz eines Filmes in der Bibliothek und hängt sie an den Film an, der als *`eineFilmsequenz`* angegeben ist. Verwenden Sie die `removeMovieClip`- oder `unloadMovie`-Aktion oder -Methode, um einen Film zu entfernen, der mit `attachMovie` angehängt wurde.

Player

Flash 5 oder höher.

Siehe auch

„`removeMovieClip`“ auf Seite 346
„`unloadMovie`“ auf Seite 380
„`MovieClip.removeMovieClip`“ auf Seite 320
„`MovieClip.unloadMovie`“ auf Seite 323

MovieClip.duplicateMovieClip

Syntax

```
eineFilmsequenz.duplicateMovieClip(neuerName, tiefe);
```

Argumente

`neuerName` Ein eindeutiger Bezeichner für die duplizierte Filmsequenz.

`tiefe` Eine Zahl zur Angabe der Tiefenebene, in der der Film abgelegt werden soll.

Beschreibung

Methode; erstellt während der Wiedergabe des Filmes eine Instanz der angegebenen Filmsequenz. Die Wiedergabe duplizierter Filmsequenzen startet immer mit Bild 1, und zwar unabhängig davon, bei welchem Bild sich die ursprüngliche Filmsequenz befindet, wenn die `duplicateMovieClip`-Methode aufgerufen wird. In der übergeordneten Filmsequenz enthaltene Variablen werden nicht in die duplizierte Filmsequenz kopiert. Beim Löschen der übergeordneten Filmsequenz wird auch die duplizierte Filmsequenz gelöscht. Mit `duplicateMovieClip` hinzugefügte Filmsequenzen können mit der `removeMovieClip`-Aktion oder -Methode gelöscht werden.

Player

Flash 5 oder höher.

Siehe auch

„`removeMovieClip`“ auf Seite 346
„`MovieClip.removeMovieClip`“ auf Seite 320

MovieClip.getBounds

Syntax
`eineFilmsequenz``.getBounds(`*`zielkoordinatensystem`*`);`

Argumente
`zielkoordinatensystem` Der Zielpfad für die Zeitleiste, deren Koordinatensystem Sie als Bezugspunkt verwenden möchten.

Beschreibung
Methode; gibt die kleinsten und größten Werte der Koordinaten *x* und *y* in der Filmsequenz für das Zielkoordinatensystem zurück, das im Argument angegeben ist. Das Rückgabeobjekt enthält die Eigenschaften `{xMin, xMax, yMin, yMax}`. Mit den `localToGlobal`- und `globalToLocal`-Methoden des MovieClip-Objektes können Sie die lokalen Koordinaten der Filmsequenz in Bühnenkoordinaten oder umgekehrt Bühnenkoordinaten in lokale Koordinaten umwandeln.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird `getBounds` verwendet, um im Koordinatensystem des Hauptfensters den Begrenzungsrahmen der Instanz `anyMovieClip` abzurufen.

```
myMovieClip.getBounds(._root);
```

Siehe auch
„`MovieClip.globalToLocal`“ auf Seite 315
„`MovieClip.localToGlobal`“ auf Seite 319

MovieClip.getBytesLoaded

Syntax
`eineFilmsequenz``.getBytesLoaded();`

Argumente
Keine.

Beschreibung
Methode; gibt die Anzahl der geladenen Bytes (als Stream) für das angegebene Filmsequenzobjekt zurück. Da interne Filmsequenzen automatisch geladen werden, wird bei dieser Methode derselbe Wert zurückgegeben wie bei `MovieClip.getBytesTotal`, sofern sich das angegebene Filmsequenzobjekt auf eine interne Filmsequenz bezieht. Diese Methode ist für die Verwendung bei geladenen Filmen gedacht. Sie können den Wert von `getBytesLoaded` mit dem Wert von `getBytesTotal` vergleichen, um zu ermitteln, zu wie viel Prozent ein externer Film geladen ist.

Player
Flash 5 oder höher.

MovieClip.getBytesTotal

Syntax
`eineFilmsequenz.getBytesTotal();`

Argumente
Keine.

Beschreibung
Methode; gibt die Größe des angegebenen Filmsequenzobjektes in Byte zurück. Bei externen Filmsequenzen (der Stammfilm oder eine Filmsequenz, die in ein Ziel oder eine Ebene geladen wird) handelt es sich beim Rückgabewert um die Größe der SWF-Datei.

Player
Flash 5 oder höher.

MovieClip.getURL

Syntax
`eineFilmsequenz.getURL(URL [,fenster, variablen]]);`

Argumente
URL Der URL, von dem das Dokument geladen wird.

fenster Ein optionales Argument für den Namen, den Frame oder den Ausdruck, mit dem das Fenster oder der HTML-Frame angegeben wird, in dem das Dokument geladen werden soll. Sie können auch einen der im Folgenden aufgeführten reservierten Zielnamen verwenden: `_self` bezeichnet den aktuellen Frame im aktuellen Fenster, `_blank` bezeichnet ein neues Fenster, `_parent` den dem aktuellen Frame übergeordneten Frame, `_top` bezeichnet den Frame der obersten Ebene des aktuellen Fensters.

variablen Ein optionales Argument, das eine Methode für das Senden von Variablen angibt, die dem zu ladenden Film zugeordnet sind. Wenn es keine Variablen gibt, lassen Sie dieses Argument aus; andernfalls geben Sie mit Hilfe der `GET`- oder `POST`-Methode an, ob Variablen geladen werden sollen. Bei `GET` werden die Variablen am Ende des URL angehängt. `GET` wird bei einer geringen Anzahl von Variablen verwendet. Bei `POST` werden die Variablen in einer separaten HTTP-Kopfzeile gesendet. `POST` wird bei langen Zeichenfolgen für Variablen verwendet.

Beschreibung
Methode; lädt ein Dokument von dem angegebenen URL in das angegebene Fenster. Mit der `getURL`-Methode können Variablen außerdem an andere Anwendungen übergeben werden, die im URL unter Verwendung der `GET`- oder `POST`-Methode definiert sind.

Player
Flash 5 oder höher.

MovieClip.globalToLocal

Syntax
eineFilmsequenz.globalToLocal(*punkt*);

Argumente
punkt Der Name oder Bezeichner eines Objektes, das mit dem generischen Object-Objekt erstellt wurde. Er gibt die *x*- und *y*-Koordinaten als Eigenschaften an.

Beschreibung
Methode; wandelt das *punkt*-Objekt von Bühnenkoordinaten (globale Koordinaten) in die Koordinaten der Filmsequenz (lokale Koordinaten) um.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel werden die globalen *x*- und *y*-Koordinaten des *punkt*-Objektes in die lokalen Koordinaten der Filmsequenz umgewandelt.

```
onClipEvent(mouseMove) {
point = new object();
point.x = _root._xmouse;
point.y = _root._ymouse;
globalToLocal(point);
_root.out = _xmouse + " === " + _ymouse;
_root.out2 = point.x + " === " + point.y;
updateAfterEvent();
}
```

Siehe auch
„MovieClip.localToGlobal“ auf Seite 319
„MovieClip.getBounds“ auf Seite 313

MovieClip.gotoAndPlay

Syntax
eineFilmsequenz.gotoAndPlay(*bild*);

Argumente
bild Die Nummer des Bilds, zu dem der Abspielkopf gesendet wird.

Beschreibung
Methode; startet die Wiedergabe des Filmes beim angegebenen Bild.

Player
Flash 5 oder höher.

MovieClip.gotoAndStop

Syntax

eineFilmsequenz.gotoAndStop(*bild*);

Argumente

bild Die Nummer des Bilds, zu dem der Abspielkopf gesendet wird.

Beschreibung

Methode; stoppt die Wiedergabe des Filmes beim angegebenen Bild.

Player

Flash 5 oder höher.

MovieClip.hitTest

Syntax

eineFilmsequenz.hitTest(*x*, *y*, *formmarkierer*);

eineFilmsequenz.hitTest(*ziel*);

Argumente

x Die *x*-Koordinate des Kollisionsbereichs auf der Bühne.

y Die *y*-Koordinate des Kollisionsbereichs auf der Bühne.

Die Koordinaten *x* und *y* sind im globalen Koordinatensystem definiert.

ziel Der Zielpfad des Kollisionsbereichs, der die von *eineFilmsequenz* angegebene Instanz eventuell überschneidet bzw. überlappt. Bei *ziel* handelt es sich in der Regel um eine Schaltfläche oder ein Textfeld.

formmarkierer Ein Boolescher Wert, der angibt, ob die gesamte Form der angegebenen Instanz (*true*) oder nur der Begrenzungsrahmen (*false*) ausgewertet werden soll. Dieses Argument kann nur angegeben werden, wenn der Kollisionsbereich durch *x*- und *y*-Koordinatenargumente bezeichnet ist.

Beschreibung

Methode; wertet die durch *eineFilmsequenz* angegebene Instanz aus und ermittelt, ob diese den Kollisionsbereich überschneidet bzw. überlappt, der durch *ziel* oder die *x*- und *y*-Koordinatenargumente bezeichnet ist.

Verwendung 1 vergleicht die Koordinaten *x* und *y* mit der Form oder dem Begrenzungsrahmen der angegebenen Instanz, und zwar entsprechend der Einstellung für *formmarkierer*. Wenn *formmarkierer* auf true gesetzt ist, wird nur der momentan von der Instanz auf der Bühne verwendete Bereich ausgewertet, und wenn *x* und *y* sich an einem beliebigen Punkt überlappen, wird der Wert true zurückgegeben. Dies erweist sich als nützlich, um festzustellen, ob sich eine Filmsequenz innerhalb eines bestimmten Kollisions- oder Hotspot-Bereich befindet.

Verwendung 2 wertet die Begrenzungsrahmen von *ziel* und der angegebenen Instanz aus und gibt `true` zurück, wenn sich diese an irgendeinem Punkt überschneiden oder überlappen.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird `hitTest` mit den Eigenschaften `x_mouse` und `y_mouse` verwendet und ermittelt, ob sich die Maus außerhalb des Begrenzungsrahmens des Ziels befindet.

```
if (hitTest( _root._xmouse, _root._ymouse, false));
```

Im folgenden Beispiel wird mit `hitTest` ermittelt, ob die Filmsequenz `ball` die Filmsequenz `square` überschneidet bzw. überlappt.

```
if(_root.ball, hittest(_root.square)){
trace("ball überschneidet square");
}
```

Siehe auch
„`MovieClip.localToGlobal`“ auf Seite 319
„`MovieClip.globalToLocal`“ auf Seite 315
„`MovieClip.getBounds`“ auf Seite 313

MovieClip.loadMovie

Syntax
`eineFilmsequenz.loadMovie(url [,variablen]);`

Argumente

Argumente
url Ein absoluter oder relativer URL für die zu ladende SWF-Datei. Ein relativer Pfad muss relativ zur SWF-Datei angegeben werden. Der URL muss sich in der gleichen Subdomäne befinden wie der URL, unter dem der Film momentan abgelegt ist. Um SWF-Dateien in Flash Player verwenden oder in der Flash Erstellungsumgebung im Filmtestmodus testen zu können, müssen diese im gleichen Ordner gespeichert sein. Die Dateinamen dürfen außerdem keine Laufwerks- und Ordnerbezeichnungen enthalten.

variablen Ein optionales Argument, das eine Methode für das Senden von Variablen angibt, die dem zu ladenden Film zugeordnet sind. Bei diesem Argument muss es sich um die Zeichenfolge „GET“ oder „POST“ handeln. Wenn es keine Variablen gibt, lassen Sie dieses Argument aus; andernfalls geben Sie mit Hilfe der `GET`- oder `POST`-Methode an, ob Variablen geladen werden sollen. Bei `GET` werden die Variablen am Ende des URL angehängt. `GET` wird bei einer geringen Anzahl von Variablen verwendet. Bei `POST` werden die Variablen in einer separaten HTTP-Kopfzeile gesendet. `POST` wird bei langen Zeichenfolgen für Variablen verwendet.

Beschreibung
Methode; gibt zusätzliche Filme wieder, ohne Flash Player zu schließen. Normalerweise zeigt der Flash Player nur einen einzelnen Flash Player-Film (SWF-Datei) an und wird dann geschlossen. Mit der `loadMovie`-Methode können Sie mehrere Filme auf einmal abspielen bzw. zwischen einzelnen Filmen hin- und herschalten, ohne ein weiteres HTML-Dokument laden zu müssen.

Mit der `unloadMovie`-Aktion können Sie Filme entfernen, die mit der `loadMovie`-Aktion geladen wurden.

Mit der `loadVariables`-Methode können Sie den aktuellen Film beibehalten und die Variablen mit neuen Werten aktualisieren.

Player
Flash 5 oder höher.

Siehe auch
„`MovieClip.loadVariables`“ auf Seite 318
„`MovieClip.unloadMovie`“ auf Seite 323

MovieClip.loadVariables

Syntax
```
eineFilmsequenz.loadVariables(url, variablen);
```

Argumente
url Der absolute oder relative URL für die externe Datei. Der Host für den URL muss sich in derselben Subdomäne wie die Filmsequenz befinden.

variablen Die Methode zum Abrufen von Variablen. Bei `GET` werden die Variablen am Ende des URL angehängt. `GET` wird bei einer geringen Anzahl von Variablen verwendet. Bei `POST` werden die Variablen in einer separaten HTTP-Kopfzeile gesendet. `POST` wird bei langen Zeichenfolgen für Variablen verwendet.

Beschreibung
Methode; liest Daten aus einer externen Datei und setzt die Werte für Variablen in einem Film oder einer Filmsequenz. Die externe Datei kann eine Textdatei sein, die von einem CGI-Skript, Active Server Pages (ASP) oder PHP erstellt wurde, und sie kann eine beliebige Anzahl an Variablen enthalten.

Mit dieser Methode können Variablen im aktiven Film auch mit neuen Werten aktualisiert werden.

Bei dieser Methode ist es erforderlich, dass der Text im URL das Standard-MIME-Format aufweist: *application/x-www-urlformencoded* (Format für CGI-Skript).

Player
Flash 5 oder höher.

Siehe auch
„`MovieClip.loadMovie`“ auf Seite 317

MovieClip.localToGlobal

Syntax

`eineFilmsequenz.localToGlobal(punkt);`

Argumente

`punkt` Der Name oder Bezeichner eines Objektes, das mit dem Object-Objekt erstellt wurde. Er gibt die *x*- und *y*-Koordinaten als Eigenschaften an.

Beschreibung

Methode; wandelt das `punkt`-Objekt von den Koordinaten der Filmsequenz (lokale Koordinaten) in die Bühnenkoordinaten (globale Koordinaten) um.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel werden die Koordinaten *x* und *y* des `punkt`-Objektes von den Koordinaten der Filmsequenz (lokale Koordinaten) in die Bühnenkoordinaten (globale Koordinaten) umgewandelt. Die lokalen Koordinaten *x* und *y* werden mit Hilfe von `xmouse` und `ymouse` angegeben, um die *x*- und *y*-Koordinaten der Mausposition abzurufen.

```
onClipEvent(mouseMove) {
point = new object();
point.x = _xmouse;
point.y = _ymouse;
_root.out3 = point.x + " === " + point.y;
_root.out = _root._xmouse + " === " + _root._ymouse;
localToGlobal(point);
_root.out2 = point.x + " === " + point.y;
updateAfterEvent();
}
```

Siehe auch

„`MovieClip.globalToLocal`“ auf Seite 315

MovieClip.nextFrame

Syntax

`eineFilmsequenz.nextFrame();`

Argumente

Keine.

Beschreibung

Methode; sendet den Abspielkopf zum nächsten Bild der Filmsequenz.

Player

Flash 5 oder höher.

MovieClip.play

Syntax
`eineFilmsequenz.play();`

Argumente
Keine.

Beschreibung
Methode; gibt die Filmsequenz wieder.

Player
Flash 5 oder höher.

MovieClip.prevFrame

Syntax
`eineFilmsequenz.prevFrame();`

Argumente
Keine.

Beschreibung
Methode; sendet den Abspielkopf zum vorigen Bild und stoppt die Wiedergabe.

Player
Flash 5 oder höher.

MovieClip.removeMovieClip

Syntax
`eineFilmsequenz.removeMovieClip();`

Argumente
Keine.

Beschreibung
Methode; entfernt eine Instanz der Filmsequenz, die mit der `duplicateMov-ieclip`-Aktion, der `duplicateMovieClip`- oder `attachMovie`-Methode des MovieClip-Objektes erstellt wurde.

Player
Flash 5 oder höher.

Siehe auch
„MovieClip.loadMovie“ auf Seite 317
„MovieClip.attachMovie“ auf Seite 311

MovieClip.startDrag

Syntax

```
eineFilmsequenz.startDrag([einrasten, links, rechts, oben,
unten]);
```

Argumente

einrasten Ein Boolescher Wert, der angibt, ob die verschiebbare Filmsequenz am Mittelpunkt der Mausposition (*true*) oder an der Stelle einrastet, an der der Benutzer zum ersten Mal auf die Filmsequenz geklickt hat (*false*). Dieses Argument ist optional.

links, oben, rechts, unten Werte, die relativ zu den Koordinaten der übergeordneten Filmsequenz sind. Mit ihnen wird ein begrenzendes Rechteck für die Filmsequenz angegeben. Diese Argumente sind optional.

Beschreibung

Methode; ermöglicht es Benutzern, die angegebene Filmsequenz mit einer Ziehoperation zu verschieben. Eine Ziehoperation ist bei dem Film so lange möglich, bis sie durch Aufruf der `stopDrag`-Methode explizit beendet oder eine Ziehoperation bei einer anderen Filmsequenz gestartet wird. Sie können eine Ziehoperation immer nur bei einer Filmsequenz durchführen.

Player

Flash 5 oder höher.

Siehe auch

„MovieClip.stopDrag“ auf Seite 321
„_droptarget“ auf Seite 261

MovieClip.stop

Syntax

```
eineFilmsequenz.stop();
```

Argumente

Keine.

Beschreibung

Methode; stoppt die Wiedergabe des momentan abgespielten Filmes.

Player

Flash 5 oder höher.

MovieClip.stopDrag

Syntax

```
eineFilmsequenz.stopDrag();
```

Argumente
Keine.

Beschreibung
Methode; beendet eine mit der `startDrag`-Methode implementierte Ziehaktion. Ein Film kann durch eine Ziehoperation verschoben werden, bis die `stopDrag`-Methode hinzugefügt oder eine Ziehoperation bei einem anderen Film gestartet wird. Sie können eine Ziehoperation immer nur bei einer Filmsequenz durchführen.

Player
Flash 5 oder höher.

Siehe auch
„_droptarget“ auf Seite 261
„MovieClip.startDrag“ auf Seite 321

MovieClip.swapDepths

Syntax
`eineFilmsequenz.swapDepths(tiefe);`

`eineFilmsequenz.swapDepths(ziel);`

Argumente
`ziel` Die Instanz der Filmsequenz, deren Tiefe mit der Instanz ausgetauscht wird, die in *`eineFilmsequenz`* angegeben ist. Beiden Instanzen muss dieselbe Filmsequenz übergeordnet sein.

`tiefe` Eine Zahl zur Angabe der Tiefenebene, in der *`eineFilmsequenz`* abgelegt werden soll.

Beschreibung
Methode; tauscht die Stapel- bzw. die *z*-Reihenfolge (Tiefenebene) der angegebenen Instanz mit dem Film, der durch das Argument *`ziel`* angegeben ist, oder dem Film, der aktuell die im Argument angegebene Ebene *`tiefe`* belegt. Beiden Filmen muss dieselbe Filmsequenz übergeordnet sein. Das Wechseln der Tiefenebenen von Filmsequenzen hat zur Folge, dass ein Film an die Position vor oder hinter den anderen Film verschoben wird. Wenn ein Film sich beim Aufrufen dieser Methode in einem Tweening-Vorgang befindet, wird der Tweening-Vorgang gestoppt.

Player
Flash 5 oder höher.

Siehe auch
„_level“ auf Seite 290

MovieClip.unloadMovie

Syntax

`eineFilmsequenz.unloadMovie();`

Argumente

Keine.

Beschreibung

Methode; entfernt eine Filmsequenz, die mit den MovieClip-Methoden `loadMovie` oder `attachMovie` geladen wurde.

Player

Flash 5 oder höher.

Siehe auch

„MovieClip.loadMovie“ auf Seite 317
„MovieClip.attachMovie“ auf Seite 311

_name

Syntax

```
instanzname._name
instanzname._name = wert;
```

Argumente

instanzname Der Instanzname einer Filmsequenz, für die die `_name`-Eigenschaft gesetzt oder abgerufen werden soll.

wert Eine Zeichenfolge, die einen neuen Instanznamen angibt.

Beschreibung

Eigenschaft; gibt den Instanznamen der Filmsequenz an.

Player

Flash 4 oder höher.

NaN

Syntax

`NaN`

Argumente

Keine.

Beschreibung

Variable; eine vordefinierte Variable mit dem im IEEE-754-Standard angegebenen Wert für NaN (Not a Number, Keine Zahl).

Player

Flash 5 oder höher.

ne (ungleich – zeichenfolgenspezifisch)

Syntax

`ausdruck1 ne ausdruck2`

Argumente

`ausdruck1, ausdruck2` Zahlen, Zeichenfolgen oder Variablen.

Beschreibung

Operator (Vergleich); vergleicht *`ausdruck1`* mit *`ausdruck2`* und gibt `true` zurück, wenn *`ausdruck1`* nicht gleich *`ausdruck2`*; andernfalls wird `false` zurückgegeben.

Player

Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen `!=`-Operators (nicht gleich) empfohlen.

Siehe auch

„`!= (nicht gleich)`“ `auf Seite 191`

new

Syntax

`new konstruktor();`

Argumente

`konstruktor` Eine Funktion gefolgt von optionalen Argumenten in Klammern. Die Funktion entspricht in der Regel dem Namen des Typs von Objekt (beispielsweise Array, Math, Number, Object), das erstellt wird.

Beschreibung

Operator; erstellt ein neues, anfangs anonymes Objekt, ruft die Funktion auf, die durch das *`konstruktor`*-Argument angegeben ist, übergibt die in Klammern angegebenen optionalen Argumente und das neu erstellte Objekt als Wert des Schlüsselworts `this`. Die Konstruktor-Funktion kann dann mit `this` das neue Objekt instantiieren.

Die _prototype_-Eigenschaft des Objektes der Konstruktor-Funktion wird in die _proto_-Eigenschaft des neuen Objektes kopiert. Aufgrund dessen unterstützt das neue Objekt sämtliche Methoden und Eigenschaften, die im prototype-Objekt der Konstruktor-Funktion angegeben sind.

Player

Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird mit dem new-Operator die Objekte `book1` und `book2` erstellt.

```
function Book(name, price)
{
  this.name = name;
  this.price = price;
}
book1 = new Book("Ignaz oder Die Verschwörung der Idioten",
19.95);
book2 = new Book("Die sizilianische Oper", 10.95);
```

Siehe auch
„[] (Arrayzugriffsoperator)“ auf Seite 202
„{} (Objektinitialisierung)“ auf Seite 204
Den Abschnitt über die Konstruktor-Methode in Einträgen zu Objekten.

newline

Syntax
`newline;`

Argumente
Keine.

Beschreibung
Konstante; fügt ein Wagenrücklauf-Zeichen ({) ein, wodurch eine leere Zeile in den ActionScript-Code eingefügt wird. Mit `newline` können Sie Platz für Informationen schaffen, die von einer Funktion oder Aktion im Code abgerufen werden.

Player
Flash 4 oder höher.

nextFrame

Syntax
`nextFrame();`

Argumente
Keine.

Beschreibung
Aktion; sendet den Abspielkopf zum nächsten Bild und stoppt die Wiedergabe.

Player
Flash 2 oder höher.

Beispiel
Wenn der Benutzer auf eine Schaltfläche klickt, der eine `nextFrame`-Aktion zugeordnet ist, wird der Abspielkopf zum nächsten Bild gesendet.

```
on (release) {
    nextFrame(5);
}
```

nextScene

Syntax
`nextScene();`

Argumente
Keine.

Beschreibung
Aktion; sendet den Abspielkopf zu Bild 1 der nächsten Szene und stoppt die Wiedergabe.

Player
Flash 2 oder höher.

Beispiel
Diese Aktion ist einer Schaltfläche zugeordnet. Beim Klicken auf diese Schaltfläche wird der Abspielkopf zum Bild 1 der nächsten Szene gesendet.

```
on(release) {
    nextScene();
}
```

not

Syntax
`not` *`ausdruck`*

Argumente
`ausdruck` Eine beliebige Variable oder ein anderer Ausdruck, der in einen Booleschen Wert umgewandelt werden kann.

Beschreibung
Operator; führt eine logische NICHT-Operation im Flash 4 Player aus.

Player
Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen `!`-Operators (logisches NICHT) empfohlen.

Siehe auch
„`! (logisches NICHT)`“ auf Seite 191

null

Syntax

`null`

Argumente

Keine.

Beschreibung

Schlüsselwort; ein spezieller Wert, der Variablen zugewiesen oder von einer Funktion anstelle von Daten zurückgegeben werden kann. Mit `null` können Sie fehlende Werte oder Werte ohne definierten Datentyp darstellen.

Player

Flash 5 oder höher.

Beispiel

Im numerischen Kontext steht `null` für 0. Mit `null` kann auf Gleichheit überprüft werden. In dieser Anweisung wird einem Knoten des binären Baumes kein linkes Unterelement hinzugefügt. Das Feld für das linke Unterelement kann daher auf `null` gesetzt werden.

```
if (tree.left == null) {
  tree.left = new TreeNode();
}
```

Number (Funktion)

Syntax

`Number(`*`ausdruck`*`);`

Argumente

`ausdruck` Die Zeichenfolge, Boolean oder ein anderer Ausdruck, der in eine Zahl umgewandelt werden kann.

Beschreibung

Funktion; wandelt das Argument `x` in eine Zahl um und gibt entsprechend den folgenden Regeln einen Wert zurück:

Wenn `x` eine Zahl ist, ist der Rückgabewert `x`.

Wenn es sich bei `x` um einen Boolean handelt und `x` den Wert `true` hat, wird der Wert 1 zurückgegeben. Wenn `x` den Wert `false`hat, wird 0 zurückgegeben.

Wenn `x` eine Zeichenfolge ist, versucht die Funktion `x` als Dezimalzahl mit optionalem nachgestelltem Exponenten, d. h. 1,57505e-3, zu lesen.

Wenn `x` undefiniert ist, ist der Rückgabewert 0.

Mit dieser Funktion können Flash 4-Dateien umgewandelt werden, die in die Flash 5-Erstellungsumgebung importiert wurden und veraltete Operatoren enthalten. Weitere Informationen finden Sie im Abschnitt über den `&`-Operator.

Player
Flash 4 oder höher.

Siehe auch
„Number (Objekt)“ auf Seite 328

Number (Objekt)

Beim Number-Objekt handelt es sich um ein einfaches Wrapperobjekt für den Datentyp Zahl. Das bedeutet, dass Sie einfache numerische Werte mit Hilfe der Methoden und Eigenschaften bearbeiten können, die dem Number-Objekt zugeordnet sind. Die Funktionen, die dieses Objekt bietet, entsprechen denen des Number-Objektes in JavaScript.

Um die Methoden des Number-Objektes aufrufen zu können, müssen Sie den `Number`-Konstruktor verwenden. Der Konstruktor wird nicht benötigt, wenn Sie die Eigenschaften des Number-Objektes aufrufen. In den folgenden Beispielen ist die Syntax angegeben, die für einen Aufruf der Methoden und Eigenschaften des Number-Objektes erforderlich ist:

Dies ist ein Beispiel für einen Aufruf der `toString`-Methode des Number-Objekts:

```
myNumber = new Number(1234);
myNumber.toString();
```

Gibt eine Zeichenfolge mit der binären Darstellung der Zahl 1234 zurück.

Dies ist ein Beispiel für einen Aufruf der `MIN_VALUE`-Eigenschaft (auch als Konstante bezeichnet) des Number-Objekts:

```
smallest = Number.MIN_VALUE
```

Zusammenfassung der Methoden für das Number-Objekt

Methode	Beschreibung
`toString`	Gibt ein Number-Objekt als Zeichenfolge zurück.
`valueOf`	Gibt den Grundwert eines Number-Objektes zurück.

Zusammenfassung der Eigenschaften für das Number-Objekt

Eigenschaft	Beschreibung
`MAX_VALUE`	Konstante, mit der die größte darstellbare Zahl angegeben wird (doppeltgenau nach IEEE-754). Diese Zahl ist ungefähr 1,7976931348623158e+308.
`MIN_VALUE`	Konstante, mit der die kleinste darstellbare Zahl angegeben wird (doppeltgenau nach IEEE-754). Diese Zahl ist ungefähr 5e-324.
`NaN`	Konstante, mit der der Wert für NaN (Not a Number, Keine Zahl) angegeben wird.
`NEGATIVE_INFINITY`	Konstante, mit der der Wert für negative Unendlichkeit dargestellt wird.
`POSITIVE_INFINITY`	Konstante, mit der der Wert für positive Unendlichkeit dargestellt wird. Dieser Wert entspricht dem Wert für die globale Variable Infinity.

Konstruktor für das Number-Objekt

Syntax

```
meineNummer = new Number(wert);
```

Argumente

`wert` Der numerische Wert des erstellten Number-Objektes oder ein Wert, der in eine Zahl umgewandelt werden soll.

Beschreibung

Konstruktor; erstellt ein neues Number-Objekt. Sie müssen den `Number`-Konstruktor verwenden, wenn Sie mit den `toString`- und `valueOf`-Methoden des Number-Objektes arbeiten. Der Konstruktor ist nicht erforderlich, wenn Sie die Eigenschaften des Number-Objektes verwenden. Der `new Number`-Konstruktor wird hauptsächlich als Platzhalter verwendet. Eine Instanz des Number-Objektes entspricht nicht der `Number`-Funktion, die ein Argument in einen Grundwert umwandelt.

Player

Flash 5 oder höher.

Beispiel

Der folgende Code erstellt neue Number-Objekte.

```
n1 = new Number(3.4);
n2 = new Number(-10);
```

Siehe auch

„`Number (Funktion)`“ auf Seite 327

Number.MAX_VALUE

Syntax

`Number.MAX_VALUE`

Argumente

Keine.

Beschreibung

Eigenschaft; die größte darstellbare Zahl (doppeltgenau nach IEEE-754). Diese Zahl lautet ungefähr 1,79E+308.

Player

Flash 5 oder höher.

Number.MIN_VALUE

Syntax

`Number.MIN_VALUE`

Argumente

Keine.

Beschreibung

Eigenschaft; die kleinste darstellbare Zahl (doppeltgenau nach IEEE-754). Diese Zahl ist ungefähr 5e-324.

Player

Flash 5 oder höher.

Number.NaN

Syntax

`Number.NaN`

Argumente

Keine.

Beschreibung
Eigenschaft; der Wert für `NaN` (Not A Number, Keine Zahl) nach IEEE-754.

Player
Flash 5 oder höher.

Number.NEGATIVE_INFINITY

Syntax
`Number.NEGATIVE_INFINITY`

Argumente
Keine.

Beschreibung
Eigenschaft; gibt den Wert für negative Unendlichkeit nach IEEE-754 zurück. Dieser Wert entspricht dem Wert für die globale Variable `Infinity`.

Negative Unendlichkeit ist ein spezieller numerischer Wert, der zurückgegeben wird, wenn eine mathematische Operation oder Funktion einen negativen Wert zurückgibt, der außerhalb des darstellbaren Bereichs liegt.

Player
Flash 5 oder höher.

Number.POSITIVE_INFINITY

Syntax
`Number.POSITIVE_INFINITY`

Argumente
Keine.

Beschreibung
Eigenschaft; gibt den Wert für positive Unendlichkeit nach IEEE-754 zurück. Dieser Wert entspricht dem Wert für die globale Variable `Infinity`.

Positive Unendlichkeit ist ein spezieller numerischer Wert, der zurückgegeben wird, wenn eine mathematische Operation oder Funktion einen positiven Wert zurückgibt, der außerhalb des darstellbaren Bereichs liegt.

Player
Flash 5 oder höher.

Number.toString

Syntax
`meineZahl.toString(grundzahl);`

Argumente
grundzahl Gibt die numerische Basis (von 2 bis 36) an, die bei Umwandlungen von Zahlen in Zeichenfolgen verwendet werden soll. Wenn Sie kein Argument *grundzahl* angeben, wird der Standardwert 10 verwendet.

Beschreibung
Methode; gibt das angegebene Number-Objekt (*meineZahl*) in einer Darstellung als Zeichenfolge zurück.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird die `Number.toString`-Methode verwendet, wobei 2 als Argument *grundzahl* angegeben ist:

```
myNumber = new Number (1000);
(1000).toString(2);
```

Gibt eine Zeichenfolge mit der binären Darstellung der Zahl 1000 zurück.

Number.valueOf

Syntax
`meineZahl.valueOf();`

Argumente
Keine.

Beschreibung
Methode; gibt den Grundwerttyp des angegebenen Number-Objektes zurück und wandelt das Number-Wrapperobjekt in den Grundwerttyp um.

Player
Flash 5 oder höher.

Object (Objekt)

Der generische Objekt vom Typ Object befindet sich auf oberster Ebene der Klassenhierarchie von ActionScript. Die Funktionen des generischen Object-Objektes bilden eine Untermenge der Funktionen, die das Object-Objekt in JavaScript aufweist.

Das generische Object-Objekt benötigt den Flash 5 Player.

Zusammenfassung der Methoden für das Object-Objekt

Methode	Beschreibung
`toString`	Wandelt das angegebene Objekt in eine Zeichenfolge um und gibt diese zurück.
`valueOf`	Gibt den Grundwert eines Object-Objektes zurück.

Konstruktor für das Object-Objekt

Syntax

```
new Object();
new Object(wert);
```

Argumente

wert Eine Zahl, ein Boolean oder eine Zeichenfolge, die in ein Objekt umgewandelt werden soll. Dieses Argument ist optional. Wenn Sie *wert* nicht angeben, erstellt der Konstruktor ein neues Objekt ohne definierte Eigenschaften.

Beschreibung

Konstruktor; erstellt ein neues Object-Objekt.

Player

Flash 5 oder höher.

Siehe auch

„Sound.setTransform" auf Seite 357
„Color.setTransform" auf Seite 238

Object.toString

Syntax

```
meinObjekt.toString();
```

Argumente

Keine.

Beschreibung

Methode; wandelt das angegebene Objekt in eine Zeichenfolge um und gibt diese zurück.

Player

Flash 5 oder höher.

Object.valueOf

Syntax

```
meinObjekt.valueOf();
```

Argumente

Keine.

Beschreibung

Methode; gibt den Grundwert des angegebenen Objektes zurück. Wenn das Objekt keinen Grundwert besitzt, wird das Objekt selbst zurückgegeben.

Player

Flash 5 oder höher.

onClipEvent

Syntax

```
onClipEvent(filmereignis);{
...
}
```

Argumente

Bei einem *filmereignis* handelt es sich um ein auslösendes Ereignis, das Aktionen ausführt, die einer Filmsequenzinstanz zugewiesen sind. Die folgenden Werte können als Argument *filmereignis* angegeben werden.

- `load` Die Aktion wird gestartet, sobald die Filmsequenz instantiiert und in der Zeitleiste angezeigt wird.
- `unload` Die Aktion wird im ersten Bild gestartet, nachdem die Filmsequenz aus der Zeitleiste entfernt wurde. Erst nachdem die mit dem Filmsequenzereignis `Unload` verknüpften Aktionen bearbeitet wurden, werden dem betreffenden Bild Aktionen hinzugefügt.
- `enterFrame` Die Aktion wird parallel zur Wiedergabe der einzelnen Bilder gestartet, ähnlich den Aktionen, die mit einer Filmsequenz verbunden sind. Die mit dem Filmsequenzereignis `OnEnterFrame` verknüpften Ereignisse werden erst nach den Aktionen verarbeitet, die mit den betreffenden Bildern verbunden sind.
- `mouseMove` Bei jeder Bewegung der Maus wird diese Aktion gestartet. Mit den Eigenschaften `_xmouse` und `_ymouse` können Sie die aktuelle Position der Maus ermitteln.
- `mouseDown` Die Aktion wird ausgeführt, wenn die linke Maustaste gedrückt wird.
- `mouseUp` Die Aktion wird ausgeführt, wenn die linke Maustaste losgelassen wird.

- `keyDown` Die Aktion wird ausgeführt, wenn eine Taste gedrückt wird. Mit der `Key.getCode`-Methode können Sie ermitteln, welche Taste zuletzt gedrückt wurde.
- `keyUp` Die Aktion wird beim Loslassen einer Taste ausgeführt. Mit der `Key.getCode`-Methode können Sie ermitteln, welche Taste zuletzt gedrückt wurde.
- `data` Die Aktion wird gestartet, wenn bei der `loadVariables` oder `loadMovie`-Aktion Daten empfangen werden. Wenn ein Verbindung zur `loadVariables`-Aktion besteht, tritt das Ereignis `data` nur einmal auf, und zwar sobald die letzte Variable geladen wurde. Wenn eine Verbindung zur `loadMovie`-Aktion besteht, tritt das Ereignis `data` wiederholt auf, und zwar beim Abrufen der einzelnen Datenabschnitte.

Beschreibung

Handler; löst Aktionen aus, die für eine bestimmte Instanz einer Filmsequenz definiert sind.

Player

Flash 5 oder höher.

Beispiel

Die folgende Anweisung liest das Skript aus einer externen Datei, sobald die Filmsequenzinstanz geladen und in der Zeitleiste erstmals angezeigt wird.

```
onClipEvent(load) {
    #include "meinScript.as"
}
```

Im folgenden Beispiel wird `onClipEvent`mit dem `keyDown`-Filmereignis verwendet. Das `keyDown`-Filmereignis wird in der Regel in Verbindung mit Methoden und Eigenschaften verwendet, die dem Key-Objekt zugeordnet sind. Im folgenden Skript wird mit `key.getCode` ermittelt, welche Taste der Benutzer gedrückt hat. Der Rückgabewert wird der `RIGHT`- oder `LEFT`-Eigenschaft des Key-Objektes zugeordnet und der Film entsprechend beeinflusst.

```
onClipEvent(keyDown) {
   if (Key.getCode() == Key.RIGHT) {
     }  _parent.nextFrame();
   else if (Key.getCode() == Key.LEFT){
      _parent.prevFrame();
}
```

Im folgenden Beispiel wird `onClipEvent` mit dem Filmereignis `mouseMove` verwendet. Über die `xmouse`- und `ymouse`-Eigenschaften kann die Position der Maus verfolgt werden.

```
onClipEvent(mouseMove) {
stageX=_root.xmouse;
stageY=_root.ymouse;
}
```

Siehe auch

„on(mouseEvent)“ auf Seite 336
„Key (Objekt)“ auf Seite 280
„_xmouse“ auf Seite 413
„_ymouse“ auf Seite 414

on(mouseEvent)

Syntax

```
on(mausereignis) {
anweisung;
}
```

Argumente

anweisung Die auszuführende Anweisung, wenn das *mausereignis* eintritt.

Der *mausereignis*-Aktion können folgende Argumente zugewiesen werden:

- `press` Die Maustaste wird gedrückt, solange sich der Mauszeiger über der Schaltfläche befindet.
- `release` Die Maustaste wird losgelassen, solange sich der Mauszeiger über der Schaltfläche befindet.
- `releaseOutside` Die Maustaste wird losgelassen, während der Mauszeiger sich nicht auf der Schaltfläche befindet.
- `rollOver` Der Mauszeiger wird über die Schaltfläche gezogen.
- `rollOut` Der Mauszeiger wird aus der Schaltfläche hinaus gezogen.
- `dragOver` Wenn sich der Mauszeiger über der Schaltfläche befindet, wird die Maustaste gedrückt. Die Maustaste wird gedrückt gehalten, und der Mauszeiger wird aus der Schaltfläche hinaus und dann wieder über die Schaltfläche gezogen.
- `dragOut` Die Maustaste wird gedrückt, wenn sich der Mauszeiger über der Schaltfläche befindet, und der Mauszeiger wird jetzt aus der Schalfläche gezogen.
- `keyPress („`*taste*`“)` Die angegebene *taste* wird gedrückt. Der durch *taste* angegebene Teil des Arguments wird unter Verwendung der Tastencodes angegeben, die Sie im Appendix B „Tastatureingaben und Tastencodewerte“ oder unter den in der „Zusammenfassung der Eigenschaften für das Key-Objekt“ auf Seite 280 aufgeführten Tastenkonstanten finden.

Beschreibung

Handler; gibt das Maus- oder die Tastenereignis zum Auslösen einer Aktion an.

Player

Flash 2 oder höher.

Beispiel

Im folgenden Skript wird die `startDrag`-Aktion ausgeführt, wenn die Maustaste gedrückt wird. Das konditionale Skript wird ausgeführt, wenn die Maustaste losgelassen und das Objekt abgelegt wird:

```
on(press) {
    startDrag("rabbit");
}
on(release) {
    if(getproperty("", _droptarget) == target) {
        setProperty ("rabbit", _x, _root.rabbit_x);
        setProperty ("rabbit", _y, _root.rabbit_y);
    } else {
_root.rabbit_x = getProperty("rabbit", _x);
_root.rabbit_y = getProperty("rabbit", _y);
_root.target = "pasture";
    }
    trace(_root.rabbit_y);
    trace(_root.rabbit_x);
    stopDrag();
}
```

Siehe auch

„`Key (Objekt)`“ auf Seite 280
„`onClipEvent`“ auf Seite 334

or

Syntax

`bedingung1` `or` *`bedingung2`*

Argumente

`bedingung1,2` Ein Ausdruck, der den Wert `true` oder `false` haben kann.

Beschreibung

Operator; wertet *`bedingung1`* und *`bedingung2`* aus. Wenn einer der beiden Ausdrücke `true` ist, hat der gesamte Ausdruck den Wert `true`.

Player

Flash 4 oder höher. Dieser Operator gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung des neuen `||`-Operators empfohlen.

Siehe auch

„`|| (ODER)`“ auf Seite 206

ord

Syntax
```
ord(zeichen);
```

Argumente
`zeichen` Das Zeichen, das in eine ASCII-Codenummer umgewandelt werden soll.

Beschreibung
Zeichenfolgenfunktion; wandelt Zeichen in ASCII-Codenummern um.

Player
Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der Methoden und Eigenschaften des `String`-Objektes empfohlen.

Siehe auch
„String (Objekt)“ auf Seite 366

_parent

Syntax
```
_parent.eigenschaft = x
_parent._parent.eigenschaft = x
```

Argumente
`eigenschaft` Die Eigenschaft, die für die aktuelle übergeordnete Filmsequenz angegeben wird.

`x` Der für die Eigenschaft gesetzte Wert. Dies ist ein optionales Argument. Es ist von der Eigenschaft abhängig, ob es gesetzt werden muss.

Beschreibung
Eigenschaft; gibt eine Referenz auf die Filmsequenz an oder zurück, in der die aktuelle Filmsequenz enthalten ist. Die aktuelle Filmsequenz ist die Filmsequenz, die das momentan ausgeführte Skript enthält. Verwenden Sie `_parent`, um einen relativen Pfad anzugeben.

Player
Flash 4 oder höher.

Beispiel
Im folgenden Beispiel ist die Filmsequenz `desk` der Filmsequenz `classroom` untergeordnet. Wenn in der Filmsequenz `desk` das folgende Skript ausgeführt wird, springt der Abspielkopf in der Zeitleiste der Filmsequenz `classroom` zu Bild 10.

```
_parent.gotoAndStop(10);
```

Siehe auch
„_root“ auf Seite 347
„targetPath“ auf Seite 374

parseFloat

Syntax

`parseFloat(zeichenfolge);`

Argumente

`zeichenfolge` Die Zeichenfolge, die eingelesen und in eine Gleitkommazahl umgewandelt werden soll.

Beschreibung

Funktion; wandelt eine Zeichenfolge in eine Gleitkommazahl um. Die Funktion liest die Zahlen in der Zeichenfolge aus und gibt diese zurück, bis der Parser auf ein Zeichen trifft, das nicht der Teil der Ausgangszahl ist. Wenn der Anfang der Zeichenfolge nicht als Zahl eingelesen werden kann, gibt `parseFloat` entweder `NaN` oder 0 zurück. Leerzeichen vor gültigen Ganzzahlen werden ebenso wie nachgestellte nicht numerische Zeichen ignoriert.

Player

Flash 5 oder höher.

Beispiel

In den folgenden Beispielen wird die Verwendung von `parseFloat` für die Auswertung unterschiedlicher Zahlentypen demonstriert:

`parseFloat(„-2„)` gibt `-2` zurück

`parseFloat(„2.5„)` gibt `2.5` zurück

`parseFloat(„3.5e6„)` gibt `3.5e6` oder `3500000` zurück

`parseFloat(„foobar„)` gibt `NaN` zurück

parseInt

Syntax

`parseInt(ausdruck, grundzahl);`

Argumente

`ausdruck` Die Zeichenfolge, Gleitkommazahl oder ein anderer Ausdruck, der eingelesen und in eine Ganzzahl umgewandelt werden soll.

`grundzahl` Eine Ganzzahl, welche die Grundzahl (Basis) der auszulesenden Zahl darstellt. Die zulässigen Werte liegen zwischen 2 und 36. Dieses Argument ist optional.

Beschreibung
Funktion; wandelt eine Zeichenfolge in eine Ganzzahl um. Wenn die in den Argumenten angegebene Zeichenfolge nicht in eine Zahl umgewandelt werden kann, gibt die Funktion `NaN` oder 0 zurück. Ganzahlen, die mit einer 0 beginnen, oder mit einer Grundzahl von 8 werden als Oktalzahlen interpretiert. Ganzzahlen, die mit 0x beginnen, werden als Hexadezimalzahlen interpretiert. Leerzeichen vor gültigen Ganzzahlen werden ebenso wie nachgestellte nicht numerische Zeichen ignoriert.

Player
Flash 5 oder höher.

Beispiel
In den folgenden Beispielen wird die Verwendung von `parseInt` für die Auswertung unterschiedlicher Zahlentypen demonstriert.

`parseInt("3.5")` gibt `3.5` zurück

`parseInt(„bar„)` gibt `NaN` zurück

`parseInt("4foo„)` gibt `4` zurück

Beispiel
Hexadezimalumwandlung:

`parseInt(„0x3F8„)` gibt `1016` zurück

`parseInt("3E8", 16)` gibt `1000` zurück

Binäre Umwandlung:

`parseInt("1010", 2)` gibt `10` (die dezimale Darstellung der binären Zahl 1010) zurück

Oktalzahlen-Parsing (die Oktalzahl wird durch die Grundzahl 8 gekennzeichnet):

`parseInt("777", 8)` gibt `511` (die dezimale Darstellung der Oktalzahl 777) zurück

play

Syntax
```
play();
```

Argumente
Keine.

Beschreibung
Aktion; verschiebt den Abspielkopf auf der Zeitleiste vorwärts.

Player
Flash 2 oder höher.

Beispiel

Im nachfolgenden Code wird eine `if`-Anweisung verwendet, um den Wert eines vom Benutzer eingegebenen Namens zu überprüfen. Wenn der Benutzer `Steve` eingibt, wird die `play`-Aktion aufgerufen und der Abspielkopf in der Zeitleiste nach vorn verschoben. Wenn der Benutzer nicht `Steve` eingibt, sondern irgendeine andere Eingabe vornimmt, wird der Film nicht wiedergegeben, und es wird ein Textfeld mit dem Variablennamen `alert` angezeigt.

```
stop();
if (name = "Steve") {
    play();
} else {
    alert = "Sie sind nicht Steve!";
}
```

prevFrame

Syntax

`prevFrame();`

Argumente

Keine.

Beschreibung

Aktion; sendet den Wiedergabekopf zum vorigen Bild und stoppt die Wiedergabe.

Player

Flash 2 oder höher.

Beispiel

Wenn der Benutzer auf eine Schaltfläche klickt, der eine `prevFrame`-Aktion zugeordnet ist, wird der Abspielkopf zum vorigen Bild gesendet.

```
on(release) {
    prevFrame(5);
}
```

Siehe auch

„MovieClip.prevFrame“ auf Seite 320

prevScene

Syntax

`prevScene();`

Argumente

Keine.

Beschreibung
Aktion; sendet den Abspielkopf zu Bild 1 der vorigen Szene und stoppt die Wiedergabe.

Player
Flash 2 oder höher.

Siehe auch
„`nextScene`“ auf Seite 326

print

Syntax

```
print (ziel, "bmovie");

print (ziel, "bmax");

print (ziel, "bframe");
```

Argumente
`ziel` Der Instanzname der zu druckenden Filmsequenz. In der Standardeinstellung werden alle Bilder des Filmes gedruckt. Wenn Sie nur bestimmte Bilder des Filmes drucken möchten, bestimmen Sie diese Bilder für den Druck, indem Sie in der Erstellungsumgebung an diese Bilder die Bildbezeichnungen `#P` anhängen.

`bmovie` Legt in einem Film den Begrenzungsrahmen eines bestimmten Bilds als Druckbereich für sämtliche Bilder des Filmes fest. Hängen Sie (in der Erstellungsumgebung) die Bezeichnung `#b` an, um das Bild zu bestimmen, dessen Begrenzungsrahmen Sie als Druckbereich verwenden möchten.

`bmax` Legt eine Zusammenstellung sämtlicher Begrenzungsrahmen aller druckbaren Bilder als Druckbereich fest. Geben Sie das Argument *`bmax`* an, wenn die druckbaren Bilder im Film verschiedene Größen aufweisen.

`bframe` Legt fest, dass der Begrenzungsrahmen jedes einzelnen druckbaren Bilds als Druckbereich für dieses Bild verwendet wird. Dies ändert den Druckbereich für jedes einzelne Bild und skaliert die Objekte, so dass sie in den Druckbereich passen. Verwenden Sie *`bframe`*, falls sich in den Bildern Objekte unterschiedlicher Größe befinden und jedes Objekt die Druckseite ausfüllen soll.

Beschreibung
Aktion; druckt die Filmsequenz *`ziel`* entsprechend der im Argument angegebenen Druckoptionen. Wenn Sie nur bestimmte Bilder des Zielfilmsdrucken möchten, hängen Sie die Bildbezeichnung `#P` an die Bilder an, die gedruckt werden sollen. Obwohl die Druckqualität der `print`-Aktion besser als die der `printAsBitmap`-Aktion ist, können Sie `print` nicht verwenden, um Filme mit Alpha-Transparenz oder speziellen Farbeffekten zu drucken.

Wenn Sie für den Druckbereich kein Argument angeben, wird der Standard-druckbereich durch die Bühnengröße des geladenen Filmes bestimmt. Der Film übernimmt nicht die Bühnengröße des Hauptfilmes. Sie können den Druck-bereich festlegen, indem Sie die Argumente `bmovie`, `bmax` oder `bframe` angeben.

Alle druckbaren Elemente eines Filmes müssen vollständig geladen werden, bevor der Druckvorgang beginnen kann.

Die Funktion zum Drucken von Flash Player unterstützt PostScript-Drucker und Nicht-PostScript-Drucker. Bei Nicht-PostScript-Druckern werden Vektoren in Bitmaps umgewandelt.

Player
Flash 5 oder höher.

Beispiel
Beim folgenden Beispiel werden alle druckbaren Bilder in `myMovie` gedruckt, wobei der Druckbereich durch den Begrenzungsrahmen des Bilds definiert ist, an das die Bildbezeichnung `#b` angehängt ist.

```
print("myMovie","bmovie");
```

Beim folgenden Beispiel werden alle druckbaren Bilder in `myMovie` gedruckt, wobei der Druckbereich durch den Begrenzungsrahmen jedes einzelnen Bilds vorgegeben wird.

```
print("myMovie","bframe");
```

Siehe auch
„`printAsBitmap`“ auf Seite 343

printAsBitmap

Syntax

```
printAsBitmap(ziel, "bmovie");
printAsBitmap(ziel, "bmax");
printAsBitmap(ziel, "bframe");
```

Argumente
`ziel` Der Instanzname der zu druckenden Filmsequenz. In der Standardeinstel-lung werden alle Bilder des Filmes gedruckt. Wenn Sie nur bestimmte Bilder des Filmes drucken möchten, bestimmen Sie diese Bilder für den Druck, indem Sie in der Erstellungsumgebung an diese Bilder die Bildbezeichnungen `#P` anhängen.

`bfilm` Legt in einem Film den Begrenzungsrahmen eines bestimmten Bilds als Druckbereich für sämtliche Bilder des Filmes fest. Hängen Sie (in der Erstel-lungsumgebung) die Bezeichnung `#b` an, um das Bild zu bestimmen, dessen Begrenzungsrahmen Sie als Druckbereich verwenden möchten.

bmax Legt eine Zusammenstellung sämtlicher Begrenzungsrahmen aller druckbaren Bilder als Druckbereich fest. Geben Sie das Argument *bmax* an, wenn die druckbaren Bilder im Film verschiedene Größen aufweisen.

bbild Legt fest, dass der Begrenzungsrahmen jedes einzelnen druckbaren Bilds als Druckbereich für dieses Bild verwendet wird. Dies ändert den Druckbereich für jedes einzelne Bild und skaliert die Objekte, so dass sie in den Druckbereich passen. Verwenden Sie *bframe*, falls sich in den Bildern Objekte unterschiedlicher Größe befinden und jedes Objekt die Druckseite ausfüllen soll.

Beschreibung
Aktion; druckt die Filmsequenz *ziel* als Bitmap. Mit `printAsBitmap` können Sie Filme drucken, in denen Bilder mit Objekten enthalten sind, die Transparenz und Farbeffekte verwenden. Die `printAsBitmap`-Aktion druckt mit der höchsten Auflösung, die der Drucker unterstützt. Auf diese Weise kann die bestmögliche Schärfe und Qualität erzielt werden. Um die Größe der Druckdatei für ein Bild zu berechnen, das als Bitmap gedruckt werden soll, multiplizieren Sie die Breite in Pixel mit der Höhe in Pixel und der Druckerauflösung.

Wenn der Film keine Alpha-Transparenzen oder Farbeffekte verwendet, empfiehlt es sich, die `print`-Aktion zu verwenden, um eine bessere Druckqualität zu erzielen.

In der Standardeinstellung wird der Druckbereich durch die Bühnengröße des geladenen Filmes bestimmt. Der Film übernimmt nicht die Bühnengröße des Hauptfilmes. Sie können den Druckbereich festlegen, indem Sie die Argumente *bmovie*, *bmax* oder *bframe* angeben.

Alle druckbaren Elemente eines Filmes müssen vollständig geladen werden, bevor der Druckvorgang beginnen kann.

Die Funktion zum Drucken von Flash Player unterstützt PostScript-Drucker und Nicht-PostScript-Drucker. Bei Nicht-PostScript-Druckern werden Vektoren in Bitmaps umgewandelt.

Player
Flash 5 oder höher.

Siehe auch
„print“ auf Seite 342

_quality

Syntax
```
_quality
_quality = x;
```

Argumente
x Eine Zeichenfolge, die einen der folgenden Werte angibt:

LOW   Niedrige Wiedergabequalität. Grafiken werden nicht mit Anti-Aliasing geglättet, und Bitmaps werden nicht geglättet.

MEDIUM   Mittlere Wiedergabequalität. Grafiken werden mit Anti-Aliasing in einem 2x2-Raster geglättet, Bitmaps werden hingegen nicht geglättet. Geeignet für Filme, die keinen Text enthalten.

HIGH   Hohe Wiedergabequalität. Grafiken werden mit Anti-Aliasing in einem 4x4-Raster geglättet, Bitmaps werden in statischen Filmen geglättet. Dabei handelt es sich um die Flash Standardeinstellung für die Wiedergabequalität.

BEST   Sehr hohe Wiedergabequalität. Grafiken werden mit Anti-Aliasing in einem 4x4-Raster geglättet, Bitmaps werden grundsätzlich geglättet.

Beschreibung
Eigenschaft (global); setzt die für einen Film verwendete Wiedergabequalität, oder ruft diese ab.

Player
Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird für die Wiedergabe von `oldQuality` die Option HIGH gesetzt.

```
oldQualtiy = _quality
_quality = "HIGH";
```

Siehe auch
„`_highquality`“ auf Seite 276

random

Syntax
```
random();
```

Argumente
`wert`   Die höchste Ganzzahl, für die `random` einen Wert zurückgibt.

Beschreibung
Funktion; gibt nach dem Zufallsprinzip eine Ganzzahl zwischen 0 und der Ganzzahl zurück, die im Argument *`wert`* angegeben ist.

Player
Flash 4. Diese Funktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `Math.random`-Methode empfohlen.

Beispiel
Bei der folgenden Verwendung von `random` wird der Wert 0, 1, 2, 3 oder 4 zurückgegeben.

```
random(5);
```

Siehe auch
„Math.random“ auf Seite 304

removeMovieClip

Syntax
`removeMovieClip(ziel);`

Argumente
ziel Der Zielpfad einer mit `duplicateMovieClip` erstellten Instanz der Filmsequenz oder der Instanzname einer mit der `attachMovie`- oder `duplicateMovie`-Methode erstellten Filmsequenz des MovieClip-Objektes.

Beschreibung
Aktion; löscht eine Filmsequenzinstanz, die mit der `attachMovie`- oder `duplicateMovieClip`-Methode des MovieClip-Objektes oder mit der `duplicateMovieClip`-Aktion erstellt wurde.

Player
Flash 4 oder höher.

Siehe auch
„duplicateMovieClip“ auf Seite 262
„MovieClip.duplicateMovieClip“ auf Seite 312
„MovieClip.attachMovie“ auf Seite 311
„MovieClip.removeMovieClip“ auf Seite 320

return

Syntax
`return[ausdruck];`

`return;`

Argumente
ausdruck Ein Typ, eine Zeichenfolge, eine Zahl, ein Array oder ein Objekt, das ausgewertet und als Wert der Funktion zurückgegeben werden soll. Dieses Argument ist optional.

Beschreibung
Aktion; gibt den von einer Funktion zurückgegebenen Wert an. Beim Ausführen der return-Aktion wird *ausdruck* ausgewertet und als Wert der Funktion zurückgegeben. Die return-Aktion beendet die Ausführung der Funktion. Wenn die `return`-Anweisung allein verwendet wird oder keine solche Anweisung in der Schleifenaktion vorhanden ist, wird `null` zurückgegeben.

Player
Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird die Verwendung von `return` demonstriert.

```
function sum(a, b, c){
return a + b + c;
}
```

Siehe auch

„function“ auf Seite 269

_root

Syntax

```
_root;
_root.filmsequenz;
_root.aktion;
```

Argumente

filmsequenz Der Instanzname einer Filmsequenz.

aktion Der für eine Eigenschaft gesetzte Wert. Dies ist ein optionales Argument. Es ist von der Eigenschaft abhängig, ob es gesetzt werden muss.

Beschreibung

Eigenschaft; gibt eine Referenz auf die Stammfilmzeitleiste an oder zurück. Wenn ein Film mehrere Ebenen besitzt, befindet sich die Stammfilmzeitleiste auf der Ebene, die das momentan ausgeführte Skript enthält. Wenn ein Skript der Ebene 1 beispielsweise _root auswertet, wird Ebene 1 zurückgegeben.

Die Angabe von `_root` entspricht der Schrägstrich-Syntax „/“, mit der ein absoluter Pfad in der aktuellen Ebene angegeben wird.

Player

Flash 4 oder höher.

Beispiel

Im folgenden Beispiel wird die Zeitleiste der Ebene gestoppt, die das momentan ausgeführte Skript enthält.

```
_rootl.stop();
```

Im folgenden Beispiel wird die Zeitleiste in der aktuellen Ebene zu Bild 3 verschoben.

```
_root.gotoAndStop(3);
```

Siehe auch

„_parent“ auf Seite 338
„targetPath“ auf Seite 374

_rotation

Syntax

```
instanzname._rotation
instanzname._rotation = ganzzahl
```

Argumente

ganzzahl Die Gradzahl, um die die Filmsequenz gedreht werden soll.

instanzname Die zu drehende Filmsequenz.

Beschreibung

Eigenschaft; gibt die Drehung der Filmsequenz in Grad an.

Player

Flash 4 oder höher.

Beispiel

scroll

Syntax

```
variablenname.scroll = x
```

Argumente

variablenname Der Name einer Variable, die an ein Textfeld gebunden ist.

x Die Zeilennummer der obersten sichtbaren Zeile im Textfeld. Sie können diesen Wert angeben oder den Standardwert 1 verwenden. Der Wert wird in Flash Player aktualisiert, wenn der Benutzer im Textfeld aufwärts oder abwärts blättert.

Beschreibung

Eigenschaft; steuert die Anzeige von Informationen in einem Textfeld, das an eine Variable gebunden ist. Mit der `scroll`-Eigenschaft wird festgelegt, an welcher Position im Textfeld mit der Anzeige von Daten begonnen wird. Nachdem Sie die Eigenschaft gesetzt haben, wird sie von Flash Player aktualisiert, sobald der Benutzer im Textfeld blättert. Die `scroll`-Eigenschaft ist besonders nützlich, wenn Benutzer auf einen bestimmten Abschnitt in einem längeren Text hingewiesen werden sollen, oder zum Erstellen von Textfeldern, in denen geblättert werden kann. Diese Eigenschaft kann abgerufen und geändert werden.

Player

Flash 4 oder höher.

Siehe auch

„maxscroll“ auf Seite 306

Selection (Objekt)

Das Selection-Objekt ermöglicht es Ihnen, das aktuell markierte bearbeitbare Textfeld zu setzen und zu steuern. Das aktuell markierte bearbeitbare Textfeld ist das Feld, in dem sich momentan der Cursor befindet. Indizes für Auswahlbereiche haben die Basis Null (die erste Position ist 0, die zweite Position 1 usw.).

Für das Selection-Objekt gibt es keine Konstruktor-Methode, da zu jedem Zeitpunkt immer nur ein Feld markiert sein kann.

Zusammenfassung der Methoden für das Selection-Objekt

Methode	Beschreibung
`getBeginIndex`	Gibt den Index am Anfang eines Auswahlbereichs zurück. Gibt -1 zurück, wenn kein Index vorhanden oder kein Feld aktiviert ist.
`getCaretIndex`	Gibt den aktuellen Einfügepunkt im aktuell markierten Auswahlbereich zurück. Gibt -1 zurück, wenn kein Einfügepunkt oder aktuell markierter Auswahlbereich vorhanden ist.
getEndIndex	Gibt den Index am Ende des Auswahlbereichs zurück. Gibt -1 zurück, wenn kein Index vorhanden oder kein Feld aktiviert ist.
`getFocus`	Gibt den Namen der Variablen für das aktuell markierte bearbeitbare Textfeld zurück. Gibt null zurück, wenn kein aktuell markiertes bearbeitbares Textfeld vorhanden ist.
`setFocus`	Aktiviert das bearbeitbare Textfeld, das der im Argument angegebenen Variablen zugeordnet ist.
`setSelection`	Setzt die Anfangs- und Endindizes des Auswahlbereichs.

Selection.getBeginIndex

Syntax

```
Selection.getBeginIndex();
```

Argumente

Keine.

Beschreibung

Methode; gibt den Index am Anfang des Auswahlbereichs zurück. Wenn kein Index vorhanden oder gerade kein Feld markiert ist, gibt die Methode -1 zurück. Indizes von Auswahlbereichen haben die Basis Null (die erste Position ist 0, die zweite Position 1 usw.).

Player

Flash 5 oder höher.

Selection.getCaretIndex

Syntax

```
Selection.getCaretIndex();
```

Argumente

Keine.

Beschreibung

Methode; gibt den Index der Position des Cursors zurück. Wenn kein Cursor angezeigt wird, gibt die Methode -1 zurück. Indizes von Auswahlbereichen haben die Basis Null (die erste Position ist 0, die zweite Position 1 usw.).

Player

Flash 5 oder höher.

Selection.getEndIndex

Syntax

```
Selection.getEndIndex();
```

Argumente

Keine.

Beschreibung

Methode; gibt den Endindex des aktuell markierten Auswahlbereichs zurück. Wenn kein Index vorhanden oder Auswahlbereich aktuell markiert ist, gibt die Methode -1 zurück. Indizes von Auswahlbereichen haben die Basis Null (die erste Position ist 0, die zweite Position 1 usw.).

Player

Flash 5 oder höher.

Selection.getFocus

Syntax

```
Selection.getFocus();
```

Argumente

Keine.

Beschreibung

Methode; gibt den Namen der Variablen für das aktuell markierte bearbeitbare Textfeld zurück. Wenn kein Textfeld aktuell markiert ist, gibt die Methode `null` zurück

Player

Flash 5 oder höher.

Beispiel

Der folgende Code gibt den Namen der Variablen zurück.

```
_root.anyMovieClip.myTextField.
```

Selection.setFocus

Syntax

```
Selection.setFocus(variable);
```

Argumente

variable Eine Zeichenfolge, die den Namen einer dem Textfeld zugeordneten Variablen in Punkt- oder Schrägstrich-Syntax zurückgibt.

Beschreibung

Methode; aktiviert das bearbeitbare Textfeld, das `variable` zugeordnet ist.

Player

Flash 5 oder höher.

Selection.setSelection

Syntax

```
Selection.setSelection(start, ende);
```

Argumente

start Der Anfangsindex des Auswahlbereichs.

ende Der Endindex des Auswahlbereichs.

Beschreibung

Methode; setzt den Auswahlbereich des aktuell markierten bearbeitbaren Textfeldes. Der neue Auswahlbereich beginnt mit dem im Argument `start` angegebenen Index und endet mit dem im Argument `ende` angegebenen Index. Indizes für Auswahlbereiche haben die Basis Null (die erste Position ist 0, die zweite Position 1 usw.). Diese Methode hat keine Auswirkungen, wenn kein aktuell markiertes bearbeitbares Textfeld vorhanden ist.

Player

Flash 5 oder höher.

set

Syntax

```
variable = ausdruck;
set(variable, ausdruck);
```

Argumente

variable Der Name des Containers, der den Wert für das Argument *ausdruck* enthält.

ausdruck Der Wert (oder eine Phrase, die einen Wert annehmen kann), der der Variablen zugewiesen werden soll.

Beschreibung

Aktion; weist einer Variablen einen Wert zu. Eine Variable ist ein Behälter für Informationen. Der Behälter an sich bleibt dabei immer gleich, nur der Inhalt kann sich ändern. Wenn Sie beim Abspielen eines Filmes den Wert einer Variablen ändern, können Sie Informationen über die Aktionen des Benutzers aufzeichnen und speichern, die Veränderung von Werten beim Abspielen eines Filmes aufzeichnen und feststellen, ob eine bestimmte Bedingung `true` oder `false` ist.

In Variablen können entweder Ziffern oder aus Buchstaben bestehende Zeichenfolgen gespeichert werden. Jeder Film bzw. jede Filmsequenz hat einen eigenen Satz von Variablen, und jede Variable enthält einen eigenen Wert, der von den Werten der Variablen aus anderen Filmen oder Filmsequenzen unabhängig ist.

ActionScript ist eine typenlose Sprache. Das heißt, dass für Variablen nicht ausdrücklich definiert werden muss, ob sie eine Zahl oder eine Zeichenfolge enthalten. Flash liest den Datentyp je nachdem als Ganzzahl oder als Zeichenfolge.

Verwenden Sie zur Über- oder Rückgabe von Werten die `set`-Anweisung in Verbindung mit der `call`-Aktion.

Player

Flash 4 oder höher.

Beispiel

In diesem Beispiel wird eine Variable mit der Bezeichnung `orig_x_pos` gesetzt, in der die ursprüngliche Position der *x*-Achse der Filmsequenz `ship` gespeichert ist. Die Filmsequenz kann damit während des Filmes wieder zurück an ihre Ausgangsposition gesetzt werden.

```
on(release) {
    set(x_pos, getProperty ("ship", _x ));
}
```

Dies ist gleichbedeutend mit folgender Schreibweise:

```
on(release) {
   orig_x_pos = getProperty ("ship", _x );
}
```

Siehe auch

„var“ auf Seite 381
„call“ auf Seite 235

setProperty

Syntax
`setProperty(ziel, eigenschaft, ausdruck);`

Argumente
`ziel` Der Pfad zum Instanzname der Filmsequenz, deren Eigenschaft gesetzt wird.

`eigenschaft` Die zu setzende Eigenschaft.

`ausdruck` Der Wert, auf den die Eigenschaft gesetzt wird.

Beschreibung
Aktion; ändert die Eigenschaft einer Filmsequenz bei der Filmwiedergabe.

Player
Flash 4 oder höher.

Beispiel
Mit dieser Anweisung wird die `_alpha`-Eigenschaft der mit `star` bezeichneten Filmsequenz auf 30 Prozent gesetzt, wenn auf die Schaltfläche geklickt wird.

```
on(release) {
    setProperty("star", _alpha = 30);
}
```

Siehe auch
„getProperty“ auf Seite 271

Sound (Objekt)

Mit dem Sound-Objekt können Sie die Sounds für eine bestimmte Instanz einer Filmsequenz oder für die globale Zeitleiste setzen und steuern, wenn Sie beim Erstellen eines neuen Sound-Objektes für *`ziel`* keine Angaben machen. Sie müssen den Konstruktor `new Sound` verwenden und eine Instanz des Sound-Objektes erstellen, bevor Sie die Methoden des Sound-Objektes aufrufen können.

Das Sound-Objekt wird nur vom Flash 5 Player unterstützt.

Zusammenfassung der Methoden für das Sound-Objekt

Methode	Beschreibung
`attachSound`	Fügt den im Argument angegebenen Sound hinzu.
`getPan`	Gibt den Wert des letzten Aufrufs von setPan zurück.
`getTransform`	Gibt den Wert des letzten Aufrufs von setTransform zurück.
`getVolume`	Gibt den Wert des letzten Aufrufs von setVolume zurück.
`setPan`	Setzt die Balance (links/rechts) des Sounds.
`setTransform`	Setzt die Transformation eines Sounds.
`setVolume`	Setzt die Lautstärke eines Sounds.
`start`	Startet die Wiedergabe eines Sounds, entweder von Anfang an oder optional von einem Offset-Punkt, der im Argument gesetzt wird.
`stop`	Stoppt die Wiedergabe des angegebenen Sound oder sämtlicher momentan abgespielten Sounds.

Konstruktor für das Sound-Objekt

Syntax

```
new Sound();

new Sound(ziel);
```

Argumente

`ziel` Die Filmsequenzinstanz, auf die sich das Sound-Objekt bezieht. Dieses Argument ist optional.

Beschreibung

Methode; erstellt ein neues Sound-Objekt für eine angegebene Filmsequenz. Wenn Sie `ziel` nicht angeben, steuert das Sound-Objekt alle Sounds in der globalen Zeitleiste.

Player

Flash 5 oder höher.

Beispiel

```
GlobalSound = new Sound();
MovieSound = new Sound(mymovie);
```

Sound.attachSound

Syntax

meinSound.attachSound("*idName*");

Argumente

idName Der Name für die neue Instanz des Sounds. Dies ist der gleiche Name, der im Dialogfeld „Eigenschaften Symbolverknüpfung" für den Bezeichner angegeben wurde. Dieses Argument muss in " " (Anführungszeichen) angegeben werden.

Beschreibung

Methode; fügt dem angegebenen Sound-Objekt den im Argument *idName* angegebenen Sound hinzu. Der Sound muss sich in der Bibliothek des aktuellen Filmes befinden und im Dialogfeld „Eigenschaften Symbolverknüpfung" zum Exportieren freigegeben sein. Um die Wiedergabe des Sounds zu starten, müssen Sie Sound.start aufrufen.

Player

Flash 5 oder höher.

Siehe auch

„Sound.start" auf Seite 361

Sound.getPan

Syntax

meinSound.getPan();

Argumente

Keine.

Beschreibung

Methode; gibt die Balance, die beim letzten Aufruf von setPan gesetzt wurde, als Ganzzahl zwischen -100 und 100 zurück. Über die Balanceeinstellung wird in einem Film die Links-Rechts-Wiedergabe des aktuellen und der zukünftigen Sounds gesteuert.

Diese Methode und die setVolume- und setTransform-Methoden sind kumulativ.

Player

Flash 5 oder höher.

Siehe auch

„Sound.setPan" auf Seite 356
„Sound.setTransform" auf Seite 357

Sound.getTransform

Syntax
meinSound.getTransform();

Argumente
Keine.

Beschreibung
Methode; gibt die Informationen zur Soundtransformation für das angegebene Sound-Objekt zurück, die beim letzten Aufruf von `setTransform` gesetzt wurden.

Player
Flash 5 oder höher.

Siehe auch
„Sound.setTransform“ auf Seite 357

Sound.getVolume

Syntax
meinSound.getVolume();

Argumente
Keine.

Beschreibung
Methode; gibt die Lautstärke als Ganzzahl von 0 bis 100 zurück, wobei 0 für Stummschaltung und 100 für volle Lautstärke steht. Die Standardeinstellung lautet 100.

Player
Flash 5 oder höher.

Siehe auch
„Sound.setVolume“ auf Seite 360

Sound.setPan

Syntax
meinSound.setPan(*bal*);

Argumente
bal Eine Ganzzahl, mit der die Balance (links/rechts) für einen Sound angegeben wird. Der Bereich der zulässigen Werte liegt zwischen -100 und 100. Bei -100 wird nur der linke Kanal, bei 100 nur der rechte Kanal verwendet. Bei 0 wird der Sound gleichmäßig über beide Kanäle übertragen.

Beschreibung
Methode; legt fest, wie der Sound über den linken und rechten Kanal (Lautsprecher) wiedergegeben wird. Bei Monosounds beeinflusst *bal*, welcher Lautsprecher (links oder rechts) den Sound wiedergibt.

Diese Methode und die `setVolume`- und `setTransform`-Methoden sind kumulativ. Beim Aufrufen der Methode werden die geltenden Einstellungen für `setPan` und `setTransform` gelöscht und aktualisiert.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel werden `setVolume` und `setPan` zum Steuern eines Sound-Objektes mit der Zielangabe „u2“ verwendet.

```
onClipEvent(mouseDown) {
// Erstellen eines Sound-Objektes und
s = new Sound(this);
// Anhängen eines Sounds aus der Bibliothek
s.attachSound("u2");
//Setzen der Lautstärke auf 50 %
s.setVolume(50);
//Deaktivieren der Soundwiedergabe auf dem rechten Kanal
s.setPan(-100);
//Sound mit einem Offset von 30 Sekunden 5 mal abspielen
s.start(30, 5);
```

Siehe auch
„Sound.setTransform“ auf Seite 357
„Sound.setVolume“ auf Seite 360

Sound.setTransform

Syntax
meinSound.setTransform(*soundTransformObjekt*);

Argumente
soundTransformObjekt Ein Objekt, das mit Hilfe des Konstruktors des generischen Object-Objektes erstellt wurde.

Beschreibung
Methode; setzt die Informationen zur Soundtransformation für ein Sound-Objekt. Diese Methode und die `setVolume`- und `setPan`-Methoden sind kumulativ. Beim Aufrufen der Methode werden die geltenden Einstellungen für `setPan` und `setVolume` gelöscht und aktualisiert. Dieser Methodenaufruf ist für erfahrene Benutzer gedacht, die Sounds spektakuläre Effekte hinzufügen möchten.

Sounddateien benötigen erheblichen Speicherplatz auf der Festplatte und im Hauptspeicher. Da Stereosounds die doppelte Kapazität gegenüber Monosounds benötigen, empfiehlt sich i. A. die Verwendung von 6-Bit-Monosounds mit 22 kHz. Mit der `setTransform`-Methode können Sie Monosounds stereo und Stereosounds mono abspielen und Sounds interessante Effekte hinzufügen.

Im Argument `soundTransformObjekt` geben Sie ein Objekt an, das Sie mit der Konstruktor-Methode des generischen Object-Objektes erstellen. Dabei verwenden Sie Parameter, die die Aufteilung des Sounds auf den rechten und linken Kanal (Lautsprecher) angeben.

Die Parameter für das Sound Transform-Objekt sind wie folgt definiert:

`ll` Ein Prozentwert, der angibt, welcher Anteil des Eingangs für den linken Kanal auf dem linken Lautsprecher wiedergegeben wird (von -100 bis 100).

`lr` Ein Prozentwert, der angibt, welcher Anteil des Eingangs für den rechten Kanal auf dem linken Lautsprecher wiedergegeben wird (von -100 bis 100).

`rr` Ein Prozentwert, der angibt, welcher Anteil des Eingangs für den rechten Kanal auf dem rechten Lautsprecher wiedergegeben wird (von -100 bis 100).

`rl` Ein Prozentwert, der angibt, welcher Anteil des Eingangs für den linken Kanal auf dem rechten Lautsprecher wiedergegeben wird (von -100 bis 100).

Das Gesamtergebnis der Parametereinstellungen wird durch folgende Formel ausgedrückt:

Ausgang links = Eingang links * ll + Eingang rechts * lr

Ausgang rechts= Eingang rechts * rr + Eingang links * rl

Die Werte für den linken oder rechten Eingang richten sich nach der Art des Sounds (stereo oder mono) im Film.

Bei Stereosounds wird der Soundeingang gleichmäßig auf den linken und den rechten Kanal verteilt, und es gelten die folgenden Standardeinstellung für die Transformation:

```
ll = 100
lr = 0
rr = 100
rl = 0
```

Bei Monosounds werden sämtliche Eingangssignale auf dem linken Lautsprecher wiedergegeben, und es gelten die folgenden Standardeinstellungen für die Transformation:

```
ll = 100
lr = 100
rr = 0
rl = 0
```

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel wird ein Sound Transform-Objekt erstellt, bei dem sowohl der linke als auch der rechte Kanal über den linken Kanal wiedergegeben werden.

```
mySoundTransformObject = new Object
mySoundTransformObject.ll = 100
mySoundTransformObject.lr = 100
mySoundTransformObject.rr = 0
mySoundTransformObject.rl = 0
```

Mit dem oben angegebenen Code wird ein Sound Transform-Objekt erstellt. Um das Objekt einem Sound-Objekt zuzuweisen, müssen Sie es mit `setTransform` folgendermaßen an das Sound-Objekt übergeben:

```
mySound.setTransform(mySoundTransformObject);
```

Die folgenden Beispiele zeigen Einstellungen, die mit `setTransform` gesetzt werden können, aber nicht mit `setVolume` oder `setPan` und auch nicht mit einer Kombination dieser beiden.

Durch diesen Code wird sowohl der linke aus auch der rechte Kanal auf dem linken Kanal wiedergegeben:

```
mySound.setTransform(soundTransformObjectLeft);
```

Im oben angegebenen Code gelten für *`soundTransformObjectLeft`* die folgenden Parameter:

```
ll = 100
lr = 100
rr = 0
rl = 0
```

Beispiel
Mit diesem Code wird ein Stereosound mono wiedergegeben:

```
setTransform(soundTransformObjectMono);
```

Für den Code oben gelten *`soundTransformObjectMono`* die folgenden Parameter:

```
ll = 50
lr = 50
rr = 50
rl = 50
```

Beispiel
Dieser Code gibt den linken Kanal mit halber Kapazität wieder und fügt den Rest des linken Kanals dem rechten Kanal hinzu:

```
setTransform(soundTransformObjectHalf);
```

Im oben angegebenen Code gelten für *soundTransformObjectLeft* die folgenden Parameter:

```
ll = 50
lr = 0
rr = 100
rl = 50
```

Siehe auch
„Konstruktor für das Object-Objekt“ auf Seite 333

Sound.setVolume

Syntax
meinSound.setVolume(*lautstärke*);

Argumente
lautstärke Eine Zahl zwischen 0 und 100, mit der die Lautstärke angegeben wird. 100 steht für die größte Lautstärke, 0 für stumm. Die Standardeinstellung lautet 100.

Beschreibung
Methode; setzt die Lautstärke für das Sound-Objekt.

Diese Methode und die setPan- und setTransform-Methoden sind kumulativ.

Player
Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird die Lautstärke auf 50 % gesetzt, und der Sound wird allmählich vom linken auf den rechten Lautsprecher gelegt.

```
onClipEvent (load) {
   i = -100;
   s = new sound();
   s.setVolume(50);
}
onClipEvent (enterFrame) {
   S.setPan(i++);
}
```

Siehe auch
„Sound.setPan“ auf Seite 356
„Sound.setTransform“ auf Seite 357

Sound.start

Syntax

```
meinSound.start();
meinSound.start([zweiterOffset, wiederholung]);
```

`zweiterOffset` Ein optionales Argument, mit dem Sie die Wiedergabe des Sounds ab einer bestimmten Stelle starten. Wenn Sie beispielsweise einen Sound mit einer Länge von 30 Sekunden haben und die Wiedergabe in der Mitte beginnen möchte, geben Sie 15 für das Argument `zweiterOffset` an. Der Sound wird nicht um 15 Sekunden verzögert, sondern erst ab der 15-Sekunden-Marke wiedergegeben.

`wiederholung` Ein optionales Argument, mit dem Sie die Anzahl der Wiederholungen für den Sound festlegen können.

Beschreibung

Methode; startet die Wiedergabe des zuletzt angefügten Sound, und zwar von Anfang an, falls kein Argument angegeben wurde, oder von der Stelle, die im Argument `zweiterOffset` angegeben ist.

Player

Flash 5 oder höher.

Siehe auch

„Sound.setPan" auf Seite 356
„Sound.stop" auf Seite 361

Sound.stop

Syntax

```
meinSound.stop();
meinSound.stop(["idName"]);
```

Argumente

`idName` Ein optionales Argument, mit dem die Wiedergabe eines bestimmten Sounds gestoppt wird. Das Argument `idName` muss in Anführungszeichen (" ") angegeben werden.

Beschreibung

Methode; stoppt alle momentan wiedergegebenen Sounds, wenn kein Argument angegeben ist, oder den im Argument `idName` angegebenen Sound.

Player

Flash 5 oder höher.

Siehe auch

„Sound.start" auf Seite 361

_soundbuftime

Syntax

```
_soundbuftime = ganzzahl;
```

Argumente

ganzzahl Die Sekunden bis zum Beginn des Streamings des Filmes.

Beschreibung

Eigenschaft (global); legt fest, wie viele Sekunden eines Sounds beim Streaming gepuffert werden sollen. Die Standardeinstellung ist 5 Sekunden.

Player

Flash 4 oder höher.

startDrag

Syntax

```
startDrag(ziel);
```

```
startDrag(ziel,[einrasten]);
```

```
startDrag(ziel [,einrasten [,links , oben , rechts, unten]]);
```

Argumente

ziel Der Zielpfad für die Filmsequenz, die gezogen werden soll.

einrasten Ein Boolescher Wert, der angibt, ob die verschiebbare Filmsequenz am Mittelpunkt der Mausposition (*true*) oder an der Stelle einrastet, an der der Benutzer zum ersten Mal auf die Filmsequenz geklickt hat (*false*). Dieses Argument ist optional.

links, oben, rechts, unten Werte, die relativ zu den Koordinaten der übergeordneten Filmsequenz sind. Mit ihnen wird ein begrenzendes Rechteck für die Filmsequenz angegeben. Diese Argumente sind optional.

Beschreibung

Aktion; gibt die unter *ziel* angegebene Filmsequenz während der Wiedergabe des Filmes für Ziehoperationen frei. Es kann jeweils nur eine Filmsequenz gezogen werden. Nach dem Ausführen der Operation `startDrag` kann die Filmsequenz so lange gezogen werden, bis die Freigabe durch die `stopDrag`-Aktion aufgehoben oder `startDrag` für eine andere Filmsequenz aufgerufen wird.

Beispiel

Wenn Sie eine Filmsequenz erstellen möchten, die Benutzer an eine beliebige Position verschieben können, weisen Sie die `startDrag`- und `stopDrag`-Aktionen einer Schaltfläche innerhalb der Filmsequenz wie im folgenden Beispiel zu:

```
on(press) {
    startDrag("",true);
}
on(release) {
    stopDrag();
}
```

Siehe auch

„`stopDrag`“ auf Seite 364

„`_droptarget`“ auf Seite 261

stop

Syntax

`stop;`

Argumente

Keine.

Beschreibung

Aktion; stoppt die Wiedergabe der momentan abgespielten Filmsequenz. Diese Aktion wird häufig verwendet, wenn eine Filmsequenz über Schaltflächen gesteuert werden soll.

Player

Flash 3 oder höher.

stopAllSounds

Syntax

`stopAllSounds();`

Argumente

Keine.

Beschreibung

Aktion; stoppt sämtliche zurzeit in einem Film wiedergegebenen Sounds, ohne jedoch den Abspielkopf anzuhalten. Sobald der Abspielkopf Bilder abspielt, in denen Streaming-Sounds enthalten sind, setzt die Wiedergabe dieser Sounds wieder ein.

Player

Flash 3 oder höher.

Beispiel
Der folgende Code kann einer Schaltfläche zugewiesen werden. Wenn ein Benutzer auf die Schaltfläche klickt, werden sämtliche Sounds im Film gestoppt.

```
on(release) {
    stopAllSounds();
}
```

Siehe auch
„Sound (Objekt)“ auf Seite 353

stopDrag

Syntax
`stopDrag();`

Argumente
Keine.

Beschreibung
Aktion; beendet die aktuelle Ziehoperation.

Player
Flash 4 oder höher.

Beispiel
Die folgende Anweisung beendet die Ziehoperation für die Instanz `mc`, sobald der Benutzer die Maustaste loslässt.

```
on(press) {
   startDrag("mc");
}
on(release) {
 stopdrag();
}
```

Siehe auch
„startDrag“ auf Seite 362
„_droptarget“ auf Seite 261

String (Funktion)

Syntax
`String(`*`ausdruck`*`);`

Argumente
`ausdruck` Zahl, Boolean, Variable oder Objekt, das in eine Zeichenfolge umgewandelt werden soll.

Beschreibung
Funktion; gibt das angegebene Argument in einer Darstellung als Zeichenfolge zurück:

Wenn *x* ein Boolean ist, wird die Zeichenfolge `true` oder `false` zurückgegeben.

Wenn *x* eine Zahl ist, wird als Zeichenfolge die Zahl in Dezimaldarstellung zurückgegeben.

Wenn *x* eine Zeichenfolge ist, wird `x` zurückgegeben.

Wenn *x* ein Objekt ist, wird als Zeichenfolge ein Wert zurückgegeben, der durch Aufruf der String-Eigenschaft des Objektes oder, wenn keine solche Eigenschaft definiert ist, durch den Aufruf von `object.toString` erstellt wurde.

Wenn *x* eine Filmsequenz ist, ist der Zielpfad der Filmsequenz in Schrägstrich-Syntax (/) der Rückgabewert.

Wenn *x* undefiniert ist, wird eine leere Zeichenfolge zurückgegeben.

Player
Flash 3 oder höher.

Siehe auch
„Object.toString“ auf Seite 333
„Number.toString“ auf Seite 332
„String (Objekt)“ auf Seite 366
„" " (Zeichenfolgentrennzeichen)“ auf Seite 365

" " (Zeichenfolgentrennzeichen)

Syntax
`"text"`

Argumente
text Beliebiger Text.

Beschreibung
Zeichenfolgentrennzeichen; geben Sie dieses Zeichen vor und nach einer Zeichenfolge an, wenn es sich um Literalzeichenfolgen und nicht um eine Variable, einen numerischen Wert oder ein anderes ActionScript-Element handelt.

Player
Flash 4 oder höher.

Beispiel
In der folgenden Anweisung wird durch Anführungszeichen festgelegt, dass es sich bei der Zeichenfolge „Prince Edward Island“ um einen Text und nicht um den Wert einer Variablen handelt:

```
province = "Prince Edward Island"
```

Siehe auch

„String (Objekt)“ auf Seite 366
„String (Funktion)“ auf Seite 364

String (Objekt)

Beim String-Objekt handelt es sich um einen Wrapper für den Grunddatentyp Zeichenfolge, mit dem Sie Methoden und Eigenschaften des String-Objektes zur Umwandlung und Bearbeitung von Werten im Grunddatentyp Zeichenfolge aufrufen können. Mit der `String()`-Funktion können Sie den Wert jedes Objektes in eine Zeichenfolge umwandeln.

Alle Methoden des String-Objektes mit Ausnahme von `concat`, `fromCharCode`, `slice` und `substr` sind generische Methoden. Das heißt, die Methoden rufen zunächst `this.toString` auf, bevor die eigentlichen Operationen durchgeführt werden. Sie können diese Methoden daher zusammen mit Objekten verwenden, bei denen es sich nicht um String-Objekte handelt.

Sie können jede der Methoden des String-Objektes unter Verwendung der Konstruktormethode `new String` aufrufen oder eine Literalzeichenfolge als Wert angeben. Wenn Sie eine Literalzeichenfolge angeben, wandelt der ActionScript-Interpreter den Wert automatisch in ein temporäres String-Objekt um, ruft die Methode auf und verwirft das temporäre String-Objekt anschließend. Die `String.length`-Eigenschaft können Sie ebenfalls zusammen mit einer Literalzeichenfolge verwenden.

Es ist wichtig, dass Sie eine Literalzeichenfolge nicht mit einer Instanz des String-Objektes verwechseln. Im folgenden Beispiel erstellt die erste Codezeile die Literalzeichenfolge `s1`, die zweite Codezeile hingegen eine Instanz des String-Objektes `s2`.

```
s1 = "foo"
s2 = new String("foo")
```

Es empfiehlt sich, Literalzeichenfolgen zu verwenden und nur dann String-Objekte zu definieren, wenn dies erforderlich ist, da String-Objekte ein unerwartetes Verhalten aufweisen können.

Zusammenfassung der Methoden für das String-Objekt

Methode	Beschreibung
`charAt`	Gibt das Zeichen an der angegebenen Position in der Zeichenfolge zurück.
`charCodeAt`	Gibt den Wert des Zeichens an der angegebenen Position als 16-Bit-Ganzzahl zwischen 0 und 65535 zurück.
`concat`	Verbindet den Text zweier Zeichenfolgen und gibt diesen als neue Zeichenfolge zurück.
`fromCharCode`	Erstellt eine Zeichenfolge aus den in den Argumenten angegebenen Zeichen und gibt diese zurück.
`indexOf`	Durchsucht die Zeichenfolge und gibt den Index des in den Argumenten angegebenen Werts zurück. Wenn der gesuchte Wert mehrmals vorhanden ist, wird der Index des ersten Vorkommens zurückgegeben. Wenn der Wert nicht gefunden wird, wird -1 zurückgegeben.
`lastIndexOf`	Gibt das letzte Vorkommen einer Teilzeichenfolge innerhalb der Zeichenfolge vor der im Argument angegebenen Startposition oder -1 zurück, wenn die Zeichenfolge nicht gefunden wird.
`slice`	Gibt eine Teilzeichenfolge aus der Zeichenfolge als neue Zeichenfolge zurück.
`split`	Unterteilt ein String-Objekt in ein Array aus Zeichenfolgen, indem die Zeichenfolge in Teilzeichenfolgen aufgeteilt wird.
`substr`	Gibt eine bestimmte Anzahl Zeichen aus einer Zeichenfolge zurück, beginnend mit der im Argument angegebenen Position.
`substring`	Gibt die Zeichen zwischen zwei in Argumenten angegebenen Indizes als Zeichenfolge zurück.
`toLowerCase`	Wandelt die Zeichenfolge in Kleinbuchstaben um und gibt das Ergebnis zurück.
`toUpperCase`	Wandelt die Zeichenfolge in Großbuchstaben um und gibt das Ergebnis zurück.

Zusammenfassung der Eigenschaften für das String-Objekt

Eigenschaft	Beschreibung
`length`	Gibt die Länge der Zeichenfolge zurück.

Konstruktor für das String-Objekt

Syntax

```
new String(wert);
```

Argumente

wert Der Ausgangswert des neuen String-Objektes.

Beschreibung

Konstruktor; erstellt ein neues String-Objekt.

Player

Flash 5 oder höher.

Siehe auch

„String (Funktion)“ auf Seite 364
„" " (Zeichenfolgentrennzeichen)“ auf Seite 365

String.charAt

Syntax

```
meineZeichenfolge.charAt(index);
```

Argumente

index Die Position des gesuchten Zeichens in der Zeichenfolge, das zurückgegeben werden soll.

Beschreibung

Methode; gibt das Zeichen zurück, das durch *Index* angegeben wurde. Der Index des ersten Zeichens in einer Zeichenfolge lautet 0. Wenn *index* nicht zwischen 0 und `string.length` - 1 einschließlich liegt, wird eine leere Zeichenfolge zurückgegeben.

Player

Flash 5 oder höher.

String.charCodeAt

Syntax

```
meineZeichenfolge.charCodeAt(index);
```

Argumente

index Die Position des Zeichens, dessen Wert ermittelt werden soll.

Beschreibung

Methode; gibt den Wert des Zeichens zurück, das durch *index* angegeben wurde. Als Ergebnis wird eine 16-Bit-Ganzzahl von 0 bis 65535 zurückgegeben.

Diese Methode ähnelt der `String.charAt`-Methode; es wird jedoch der Wert des Zeichens an einer bestimmten Position und nicht eine Teilzeichenfolge mit dem gesuchten Zeichen zurückgegeben.

Player
Flash 5 oder höher.

String.concat

Syntax
meineZeichenfolge.concat(*wert1,...wertN*);

Argumente
wert1,...wertN Kein, ein oder mehr Werte, die verbunden werden sollen.

Beschreibung
Methode; verbindet die angegebenen Werte und gibt das Ergebnis als neue Zeichenfolge zurück. Wenn erforderlich, wird jedes Argument *Wert* zunächst in eine Zeichenfolge umgewandelt und in der Reichenfolge der Argumente an das Ende der Zeichenfolge angehängt.

Player
Flash 5 oder höher.

String.fromCharCode

Syntax
meineZeichenfolge.fromCharCode(*c1,c2,...cN*);

Argumente
c1,c2,...cN Die Zeichen, die zu einer Zeichenfolge zusammengefasst werden sollen.

Beschreibung
Methode; erstellt aus den in den Argumenten angegebenen Zeichen eine Zeichenfolge und gibt diese zurück.

Player
Flash 5 oder höher.

String.indexOf

Syntax
meineZeichenfolge.indexOf(*wert*);

meineZeichenfolge.indexOf (*wert*, *start*);

Argumente
wert Eine Ganzzahl oder eine Zeichenfolge, die die in *meineZeichenfolge* zu suchende Teilzeichenfolge enthält.

start Eine Ganzzahl, die die Startposition der Teilzeichenfolge angibt. Dieses Argument ist optional.

Beschreibung
Methode; durchsucht die Zeichenfolge und gibt die Position des ersten Vorkommens von *wert* zurück. Wenn der Wert nicht gefunden wird, gibt die Methode -1 zurück.

Player
Flash 5 oder höher.

String.lastIndexOf

Syntax
meineZeichenfolge.lastIndexOf(*teilzeichenfolge*);

meineZeichenfolge.lastIndexOf(*teilzeichenfolge*, *start*);

Argumente
teilzeichenfolge Eine Ganzzahl oder eine Zeichenfolge, die die zu suchende Zeichenfolge enthält.

start Eine Ganzzahl, die die Startposition innerhalb der Teilzeichenfolge angibt. Dieses Argument ist optional.

Beschreibung
Methode; durchsucht die Zeichenfolge und gibt den Index des letzten Vorkommens von teilzeichenfolge innerhalb der aufrufenden Zeichenfolge zurück. Wenn *teilzeichenfolge* nicht gefunden wird, gibt die Methode -1 zurück.

Player
Flash 5 oder höher.

String.length

Syntax
string.length

Argumente
Keine.

Beschreibung
Eigenschaft; gibt die Zahl der Zeichen im angegebenen String-Objekt zurück. Der Index des letzen Zeichens für eine Zeichenfolge x ist x.length-1.

Player
Flash 5 oder höher.

String.slice

Syntax

`meineZeichenfolge.slice(start, ende);`

Argumente

`start` Gibt den Index des Segmentanfangs an. Wenn `start` negativ ist, wird der Anfangspunkt ausgehend vom Ende der Zeichenfolge ermittelt, wobei -1 das letzte Zeichen der Zeichenfolge ist.

`ende` Gibt den Index für das Segmentende an. Wenn `ende` nicht angegeben wird, beinhaltet das Segment sämtliche Zeichen ab dem Wert start bis zum Ende der Zeichenfolge. Wenn `ende` negativ ist, wird der Endpunkt ausgehend vom Ende der Zeichenfolge ermittelt, wobei -1 das letzte Zeichen der Zeichenfolge ist.

Beschreibung

Methode; extrahiert ein Segment (eine Teilzeichenfolge) aus dem angegebenen String-Objekt und gibt dieses als neue Zeichenfolge zurück, ohne das ursprüngliche String-Objekt zu verändern. Die zurückgegebene Zeichenfolge umfasst das `start`-Zeichen und alle Zeichen bis (aber nicht einschließlich) zum `ende`-Zeichen.

Player

Flash 5 oder höher.

String.split

Syntax

`meineZeichenfolge.split(trennzeichen);`

Argumente

`trennzeichen` Das Zeichen, das als Trennzeichen in der Zeichenfolge verwendet wurde.

Beschreibung

Methode; unterteilt ein String-Objekt, indem die Zeichenfolge bei jedem Vorkommen von `trennzeichen` getrennt wird, und gibt die Teilzeichenfolgen als Array zurück. Wenn kein Trennzeichen angegeben wird, enthält das zurückgegebene Array nur die ursprüngliche Zeichenfolge in einem Element, d. h. die Zeichenfolge selbst. Wenn als Trennzeichen eine leere Zeichenfolge angegeben wird, wird jedes Zeichen des String-Objektes als Element in das Array aufgenommen.

Player

Flash 5 oder höher.

String.substr

Syntax

`meineZeichenfolge.substr(start, länge);`

Argumente

`start` Eine Ganzzahl, die die Position des ersten Zeichens angibt, das in die Teilzeichenfolge aufgenommen werden soll. Wenn `start` negativ ist, wird der Anfangspunkt ausgehend vom Ende der Zeichenfolge ermittelt, wobei -1 das letzte Zeichen der Zeichenfolge ist.

`länge` Die Anzahl der in die Teilzeichenfolge aufzunehmenden Zeichen. Wenn `länge` nicht angegeben wird, enthält die Teilzeichenfolge sämtliche Zeichen vom Wert Start bis zum Ende der Zeichenfolge.

Beschreibung

Methode; gibt die Zeichen in einer Zeichenfolge beginnend beim Argument `start` bis zu der im Argument `länge` angegebenen Länge zurück.

Player

Flash 5 oder höher.

String.substring

Syntax

`meineZeichenfolge.substring(von, bis);`

Argumente

`von` Eine Ganzzahl, die die Position des ersten Zeichens für die zu erstellende Teilzeichenfolge angibt. Gültige Werte für `von` sind 0 bis `string.length` - 1.

`bis` Eine Ganzzahl, die 1 + den Index des letzten Zeichens für die zu erstellende Teilzeichenfolge angibt. Gültige Werte für `von` sind 1 bis `string.length` - 1. Wenn das Argument `bis` nicht angegeben wird, wird als Ende der Teilzeichenfolge das Ende der Zeichenfolge angenommen. Wenn `von` und `bis` identisch sind, gibt die Methode eine leere Zeichenfolge zurück. Wenn `von` größer als `bis` ist, werden die Argumente vor dem Ausführen der Funktion automatisch vertauscht.

Beschreibung

Methode; gibt eine Zeichenfolge zurück, die alle Zeichen zwischen den Positionen `von` und `bis` enthält.

Player

Flash 5 oder höher.

String.toLowerCase

Syntax
`meineZeichenfolge.toLowerCase();`

Argumente
Keine.

Beschreibung
Methode; gibt eine Kopie des String-Objektes zurück, in der alle Großbuchstaben in Kleinbuchstaben umgewandelt wurden.

Player
Flash 5 oder höher.

String.toUpperCase

Syntax
`meineZeichenfolge.toUpperCase();`

Argumente
Keine.

Beschreibung
Methode; gibt eine Kopie des String-Objektes zurück, in der alle Kleinbuchstaben in Großbuchstaben umgewandelt wurden.

Player
Flash 5 oder höher.

substring

Syntax
`substring(zeichenfolge, index, anzahl);`

Argumente
zeichenfolge Die Zeichenfolge, aus der die neue Zeichenfolge extrahiert werden soll.

index Die Nummer des ersten zu extrahierenden Zeichens.

anzahl Die Anzahl der Zeichen ohne das Indexzeichen, die die extrahierte Zeichenfolge umfassen soll.

Beschreibung
Zeichenfolgenfunktion; extrahiert einen Teil der Zeichenfolge.

Player
Flash 4 oder höher. Diese Funktion gilt in Flash 5 als veraltet.

Siehe auch
„String.substring“ auf Seite 372

_target

Syntax
instanzname._target

Argumente
instanzname Der Name der Instanz einer Filmsequenz.

Beschreibung
Eigenschaft (schreibgeschützt); gibt den Zielpfad für die im Argument *instanzname* angegebene Filmsequenzinstanz zurück.

Player
Flash 4 oder höher.

targetPath

Syntax
targetpath(*filmsequenzObjekt*);

Argumente
filmsequenzObjekt Referenz (zum Beispiel _root oder _parent) für die Filmsequenz, deren Zielpfad abgefragt wird.

Beschreibung
Funktion; gibt in einer Zeichenfolge den Zielpfad für *filmsequenzObjekt* zurück. Der Zielpfad wird in Punkt-Syntax zurückgegeben. Wenn Sie den Zielpfad in Schrägstrich-Syntax abrufen möchten, verwenden Sie die _target-Eigenschaft.

Player
Flash 5 oder höher.

Beispiel
Die folgenden Beispiele sind äquivalent. Im ersten Beispiel wird die Punkt-Syntax und im zweiten Beispiel die Schrägstrich-Syntax verwendet.

```
targetPath (Board.Block[index*2+1]) {
play();
}
```

Dies entspricht:

```
tellTarget ("Board/Block:" + (index*2+1)) {
play();
}
```

Siehe auch
„eval“ auf Seite 264

tellTarget

Syntax

```
tellTarget(ziel) {
anweisung;
}
```

Argumente

ziel Eine Zeichenfolge mit dem Zielpfad für die zu steuernde Zeitleiste.

anweisung Die für die ausgewählte Zeitleiste auszuführenden Anweisungen.

Beschreibung

Aktion; führt die im Argument *anweisungen* angegebenen Anweisungen für die im Argument *ziel* angegebene Zeitleiste aus. Die `tellTarget`-Aktion eignet sich für Navigationselemente. Weisen Sie `tellTarget` Schaltflächen zu, die Filmsequenzen an einer anderen Position auf der Bühne abspielen oder anhalten. Auf diese Weise können Sie auch eine Filmsequenz zu einem bestimmten Bild springen lassen. Weisen Sie `tellTarget` beispielsweise Schaltflächen zu, über die Filmsequenzen auf der Bühne abgespielt oder angehalten oder an eine bestimmte Bildposition gesetzt werden.

Die `tellTarget`-Aktion ähnelt der `with`-Aktion, wobei jedoch mit `with` sowohl auf Filmsequenzen als auch auf andere Objekte verwiesen werden kann. `tellTarget` erfordert die Angabe des Zielpfads einer Filmsequenz und kann keine Objekte steuern.

Player

Flash 3 oder höher. Diese Aktion gilt in Flash 5 als veraltet. Stattdessen wird die Verwendung der `with`-Aktion empfohlen.

Beispiel

Die folgende `tellTarget`-Anweisung steuert eine Filmsequenzinstanz `ball` auf der Hauptzeitleiste. Bild 1 der Filmsequenz ist leer und enthält die `stop`-Aktion. Auf der Bühne wird sie daher nicht angezeigt. Sobald auf die Schaltfläche mit der folgenden Aktion geklickt wird, setzt `tellTarget` den Abspielkopf der Filmsequenz `ball` auf Bild 2 und spielt die hier beginnende Animation ab.

```
on(release) {
    tellTarget("ball") {
        gotoAndPlay(2);
    }
}
```

Siehe auch

„with“ auf Seite 385

this

Syntax
`this`

Argumente
Keine.

Beschreibung
Schlüsselwort; verweist auf ein Objekt oder eine Filmsequenzinstanz. Das Schlüsselwort `this` erfüllt in ActionScript dieselbe Aufgabe wie in JavaScript, weist jedoch einige zusätzliche Funktionen auf. In ActionScript verweist `this` beim Ausführen eines Skripts auf die Filmsequenzinstanz, die das Skript beinhaltet. Wenn `this` beim Aufruf einer Methode verwendet wird, enthält es eine Referenz auf das Objekt mit der ausgeführten Methode.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel verweist das Schlüsselwort `this` auf das Circle-Objekt.

```
function Circle(radius){
  this.radius = radius;
  this.area = math.PI * radius * radius;
}
```

In der folgenden Anweisung, die einem Bild zugewiesen ist, verweist das Schlüsselwort `this` auf die aktuelle Filmsequenz.

```
//Setzen der alpha-Eigenschaft der aktuellen Filmsequenz auf 20.

this._alpha = 20;
```

In der folgenden Anweisung innerhalb eines `onClipEvent`-Handlers verweist das Schlüsselwort `this` auf die aktuelle Filmsequenz.

```
//Beim Laden der Filmsequenz wird eine startDrag-Operation für die
aktuelle Filmsequenz initiiert.

onClipEvent (load) {
  startDrag (this, true);
}
```

Siehe auch
„new“ auf Seite 324

toggleHighQuality

Syntax

`toggleHighQuality();`

Argumente

Keine.

Beschreibung

Aktion; aktiviert und deaktiviert Anti-Aliasing im Flash Player. Anti-Aliasing bewirkt die Kantenglättung von Objekten und verlangsamt die Filmwiedergabe. Die `toggleHighQuality`-Aktion wirkt sich auf alle Filme im Flash Player aus.

Player

Flash 2 oder höher.

Beispiel

Wenn Sie den folgenden Code einer Schaltfläche zuweisen, kann das Anti-Aliasing durch Klicken auf die Schaltfläche aktiviert oder deaktiviert werden.

```
on(release) {
    toggleHighQuality();
}
```

Siehe auch

„_quality“ auf Seite 344

„_highquality“ auf Seite 276

_totalframes

Syntax

instanzname`._totalframes`

Argumente

instanzname Der Name der auszuwertenden Filmsequenz.

Beschreibung

Eigenschaft (schreibgeschützt); wertet die im Argument *instanzname* angegebene Filmsequenz aus und gibt die Gesamtzahl der Bilder im Film zurück.

Player

Flash 4 oder höher.

trace

Syntax

`trace(ausdruck);`

Argumente

`ausdruck` Eine auszuwertende Anweisung. Beim Testen eines Filmes werden die Ergebnisse des Arguments *`ausdruck`* im Ausgabefenster angezeigt.

Beschreibung

Aktion; wertet *`ausdruck`* aus und zeigt die Ergebnisse im Filmtestmodus im Ausgabefenster an.

Verwenden Sie `trace`, um während der Programmierung Anmerkungen aufzuzeichnen oder beim Testen eines Filmes Meldungen im Ausgabefenster anzuzeigen. Über den Parameter *`ausdruck`* können Sie auf bestimmte Bedingungen prüfen oder Werte im Ausgabefenster anzeigen. Die `trace`-Aktion ist mit der `alert`-Funktion in JavaScript vergleichbar.

Player

Flash 4 oder höher.

Beispiel

Dieses Beispiel stammt aus einem Spiel, in dem eine verschiebbare Filmsequenzinstanz mit der Bezeichnung `rabbi` auf einem bestimmten Ziel abgelegt werden muss. Eine Bedingung wertet die `_droptarget`-Eigenschaft aus und führt in Abhängigkeit von der Position, an der `rabbi` abgelegt wurde, unterschiedliche Aktionen aus. Über die `trace`-Aktion wird am Ende des Skripts die Position der Filmsequenz `rabbi` ausgewertet und das Ergebnis im Ausgabefenster angezeigt. Wenn `rabbi` nicht wie erwartet reagiert (wenn es z. B. am falschen Ziel einrastet), kann anhand der über die `trace`-Aktion an das Ausgabefenster gesendeten Werte leichter der Fehler im Skript ermittelt werden.

```
on(press) {
rabbi.startDrag();
}
on(release) {
if(eval(_droptarget) != target) {
     rabbi._x = rabbi_x;
     rabbi._y = rabbi_y;
} else {
rabbi_x = rabbi._x;
rabbi_y = rabbi._y;
target = "_root.pasture";
}
trace("rabbi_y = " + rabbi_y);
trace("rabbi_x = " + rabbi_x);
stopDrag();
}
```

typeof

Syntax

```
typeof(ausdruck);
```

Argumente

ausdruck Ein Objekt, eine Zeichenfolge, eine Filmsequenz oder eine Funktion.

Beschreibung

Operator; ein unärer Operator, der einem einzelnen Argument vorangestellt wird. Der Operator veranlasst Flash zur Auswertung von *ausdruck*. Als Ergebnis wird in einer Zeichenfolge angegeben, ob es sich bei dem Ausdruck um eine Zeichenfolge, eine Filmsequenz, ein Objekt oder eine Funktion handelt.

Player

Flash 5 oder höher.

unescape

Syntax

```
unescape(x);
```

Argumente

x Eine Zeichenfolge, in der Hexadezimal-Sequenzen in ASCII-Sequenzen umgewandelt werden sollen.

Beschreibung

Funktion der obersten Ebene; wertet das Argument *x* als Zeichenfolge aus, dekodiert die Zeichenfolge aus einem URL-Format (wandelt alle Hexadezimal-Sequenzen in ASCII-Zeichen um) und gibt die Zeichenfolge zurück.

Player

Flash 5 oder höher.

Beispiel

Das folgende Beispiel demonstriert den Umwandlungsprozess.

```
escape("Hello{[World]}");
```

Das Ergebnis nach escape lautet wie folgt:

```
("Hello%7B%5BWorld%5D%7D");
```

Verwenden Sie `unescape`, um das Ausgangsformat wiederherzustellen:

```
unescape("Hello%7B%5BWorld%5D%7D")
```

Das Ergebnis lautet wie folgt:

```
Hello{[World]}
```

unloadMovie

Syntax

`unloadMovie(position);`

Argumente

position Die Tiefenebene oder Zielfilmsequenz, aus der der Film entladen werden soll.

Beschreibung

Aktion; entfernt einen Film aus dem Flash Player, der zuvor mit der `loadMovie`-Aktion geladen wurde.

Player

Flash 3 oder höher.

Beispiel

Im folgenden Beispiel wird der Hauptfilm entladen, woraufhin auf der Bühne keine Elemente mehr angezeigt werden.

```
unloadMovie(_root);
```

Das folgende Beispiel entlädt den Film auf Ebene 15, sobald der Benutzer die Maustaste drückt.

```
on(press) {
  unloadMovie(_level15);
}
```

Siehe auch

„loadMovie“ auf Seite 291

updateAfterEvent

Syntax

`updateAfterEvent(filmsequenzereignis);`

Argumente

filmsequenzereignis Sie können einen der folgenden Werte als Filmsequenzereignis angeben:

- `mouseMove` Bei jeder Bewegung der Maus wird diese Aktion gestartet. Mit den Eigenschaften `_xmouse` und `_ymouse` können Sie die aktuelle Position der Maus ermitteln.
- `mouseDown` Die Aktion wird ausgeführt, wenn die linke Maustaste gedrückt wird.
- `mouseUp` Die Aktion wird ausgeführt, wenn die linke Maustaste losgelassen wird.

- `keyDown` Die Aktion wird ausgeführt, wenn eine Taste gedrückt wird. Mit der `Key.getCode`-Methode können Sie ermitteln, welche Taste zuletzt gedrückt wurde.
- `keyUp` Die Aktion wird beim Loslassen einer Taste ausgeführt. Mit der `key.getCode`-Methode können Sie ermitteln, welche Taste zuletzt gedrückt wurde.

Beschreibung

Aktion; aktualisiert die Anzeige (unabhängig von der für den Film festgelegten Bildrate), nachdem das im Argument angegebene Sequenzereignis abgeschlossen ist. Diese Aktion wird nicht im Bedienfeld "Aktionen" von Flash aufgeführt. Wenn Sie die `updateAfterEvent`-Aktion zusammen mit Ziehaktionen verwenden, bei denen die `_x`- und `_y`-Eigenschaften während einer Mausbewegung gesetzt werden, können Sie Objekte kontinuierlich und ohne Flackern über den Bildschirm bewegen.

Player

Flash 5 oder höher.

Siehe auch

„`onClipEvent`“ auf Seite 334

_url

Syntax

`*instanzname*._url`

Argumente

`instanzname` Die Zielfilmsequenz.

Beschreibung

Eigenschaft (schreibgeschützt); ermittelt den URL der SWF-Datei, von dem die Filmsequenz heruntergeladen wurde.

Player

Flash 4 oder höher.

var

Syntax

```
var variablenName1 [= wert1] [...,variablenNameN [=wertN]];
```

Argumente

`variablenName` Der Name der Variable, die deklariert werden soll.

`wert` Der Wert, der der Variable zugewiesen wird.

Beschreibung

Aktion; wird für die Deklaration lokaler Variablen verwendet. Wenn Sie lokale Variablen innerhalb einer Funktion deklarieren, sind diese Variablen ausschließlich in dieser Funktion definiert und verlieren am Ende des Funktionsaufrufs ihre Gültigkeit. Wenn Variablen nicht innerhalb eines Blocks deklariert werden und der Aufruf der Aktionsliste über eine `call`-Aktion erfolgt, sind die Variablen lokal definiert und verlieren am Ende der aktuellen Liste ihre Gültigkeit. Wenn Variablen nicht innerhalb eines Blocks deklariert werden und der Aufruf der aktuellen Aktion nicht über eine call-Aktion erfolgt, sind die Variablen nicht lokal.

Player

Flash 5 oder höher.

_visible

Syntax

```
instanzname._visible
instanzname._visible = Boolean;
```

Argumente

`Boolean` Geben Sie den Wert *`true`* oder *`false`* ein, um die Filmsequenz anzeigen zu lassen oder auszublenden.

Beschreibung

Eigenschaft; legt fest, ob der unter *`instanzname`* angegebene Film sichtbar ist oder nicht. Nicht sichtbare Filmsequenzen (wenn die Eigenschaft auf `false` gesetzt ist) sind deaktiviert. So können Sie in einer Filmsequenz, deren `_visible` -Eigenschaft auf den Wert `false` gesetzt ist, nicht mehr auf darin enthaltene Schaltflächen klicken.

Player

Flash 4 oder höher.

void

Syntax

```
void (ausdruck);
```

Argumente

`ausdruck` Ein Ausdruck mit einem beliebigen Wert.

Beschreibung

Operator; ein unärer Operator, der den Wert in *`ausdruck`* verwirft und einen undefinierten Wert zurückgibt. Der `void`-Operator wird häufig verwendet, um einen URL auszuwerten und auf Nebeneffekte zu prüfen, ohne den ausgewerteten Ausdruck im Browserfenster anzuzeigen. Der `void`-Operator wird weiterhin in Vergleichsoperationen mit dem ==-Operator verwendet, um auf undefinierte Werte zu überprüfen.

Player
Flash 5 oder höher.

while

Syntax
```
while(bedingung) {
anweisung(en);
}
```

Argumente
`bedingung` Die Anweisung, die bei jedem Ausführen der `while`-Aktion neu ausgewertet wird. Wenn die Anweisung den Wert `true` hat, wird der Ausdruck unter `anweisung(en)` ausgeführt.

`anweisung(en)` Der auszuführende Ausdruck, wenn die Prüfung der Bedingung den Wert `true` ergibt.

Beschreibung
Aktion; führt wiederholt eine Anweisung oder mehrere Anweisungen in einer Schleife aus, solange die Bedingung erfüllt (`true`) ist. Am Ende jeder `while`-Aktion wird die Schleife in Flash durch Prüfen der Bedingung neu begonnen. Wenn die Bedingung den Wert `false` hat oder gleich 0 ist, springt Flash zur ersten Anweisung nach der `while`-Aktion.

Schleifen werden üblicherweise für das Ausführen einer Aktion in Abhängigkeit von einer Zählervariable verwendet, die hierbei unter einem bestimmten Wert liegen muss. Am Ende der Schleife wird der Zähler inkrementiert, bis der gewünschte Grenzwert erreicht ist. Die `bedingung` hat dann nicht mehr den Wert `true`, und die Schleife wird beendet.

Player
Flash 4 oder höher.

Beispiel

In diesem Beispiel werden fünf Filmsequenzen auf der Bühne dupliziert, wobei die *x*- und *y*-Position sowie die `xscale`-, `yscale`- und `_alpha`-Eigenschaften zufällig festgelegt werden, um die Sequenzen auf dem Bildschirm zu verteilen. Die Variable `foo` wird mit dem Wert 0 initialisiert. Das Argument *`bedingung`* wird so gesetzt, dass die `while`-Schleife fünfmal ausgeführt oder bis der Wert der Variablen `foo` kleiner als 5 ist. Innerhalb der `while`-Schleife wird eine Filmsequenz dupliziert und `setProperty` zur Anpassung der diversen Eigenschaften der duplizierten Filmsequenz verwendet. Die letzte Anweisung in der Schleife inkrementiert `foo`. Wenn der Wert 5 erreicht ist, hat das Argument *`bedingung`* den Wert `false` und die Schleife wird nicht ausgeführt.

```
on(release) {
   foo = 0;
   while(foo < 5) {
        duplicateMovieClip("/flower", "mc" + foo, foo);
        setProperty("mc" + foo, _x, random(275));
        setProperty("mc" + foo, _y, random(275));
        setProperty("mc" + foo, _alpha, random(275));
        setProperty("mc" + foo, _xscale, random(200));
        setProperty("mc" + foo, _yscale, random(200));
        foo = foo + 1;
    }
}
```

Siehe auch

„`do... while`“ auf Seite 260
„`continue`“ auf Seite 239

_width

Syntax

`instanzname``._width`
`instanzname``._width =`*`wert`*`;`

Argumente

`wert` Die Breite des Filmes in Pixel.

`instanzname` Der Instanzname der Filmsequenz, deren `_width`-Eigenschaft gesetzt oder abgefragt werden soll.

Beschreibung

Eigenschaft; setzt die Breite des Filmes. In früheren Versionen von Flash waren `_height` und `_width` schreibgeschützte Eigenschaften. In Flash 5 können diese Eigenschaften gesetzt und abgefragt werden.

Player

In Flash 4 als schreibgeschützte Eigenschaft. Ab Flash Version 5 kann diese Eigenschaft sowohl gesetzt als auch abgefragt werden.

Beispiel
Das folgende Codebeispiel setzt die Eigenschaften für Höhe und Breite einer Filmsequenz, wenn der Benutzer mit der Maus klickt.

```
onclipEvent(mouseDown) {
   _width=200;
   _height=200;
}
```

Siehe auch
„_height“ auf Seite 275

with

x209E6 | IDS_ACTIONHELP_WITH, with statement

Syntax

```
with (objekt) {
anweisung(en);
}
```

Argumente
objekt Eine Instanz eines ActionScript-Objektes oder einer Filmsequenz.

anweisung(en) Eine oder mehrere Aktionen, die in geschweifte Klammern eingeschlossen werden.

Beschreibung
Aktion; ändert zeitweilig den Gültigkeitsbereich (oder Zielpfad) für die Auswertung der Ausdrücke und Aktionen unter *anweisung(en)*. Nach dem Ausführen der `with`-Aktion wird der ursprüngliche Gültigkeitsbereich wiederhergestellt.

Das *objekt* wird zum Kontext, in dem die Eigenschaften, Variablen und Funktionen interpretiert werden. Wenn *object* beispielsweise `meinArray` lautet und zwei Eigenschaften `length` und `concat` angegeben werden, werden diese Eigenschaften automatisch als `meinArray.length` und `meinArray.concat` interpretiert. Wenn in einem anderen Beispiel *object* gleich `state.california` ist, gelten die Aktionen und Anweisungen innderhalb der `with`-Aktion wie durch die `california`-Instanz aufgerufen.

Um den Wert eines Bezeichners unter *anweisung(en)* zu ermitteln, beginnt ActionScript am Anfang des Gültigkeitsbereichs, der durch *objekt* angegeben wurde, und sucht nach einer festgelegten Reihenfolge der Gültigkeitbereiche nach diesem Bezeichner.

Der in der `with`-Aktion für das Auflösen von Bezeichnern verwendete Gültigkeitsbereich beginnt mit dem ersten und endet mit dem letzten der in der folgenden Liste aufgeführten Elemente:

- *objekt*, auf das in der innersten `with`-Aktion verwiesen wird
- *objekt*, auf das in der äußersten `with`-Aktion verwiesen wird

- Aktivierungsobjekt (ein temporäres Objekt, das automatisch beim Aufruf einer Funktion erstellt wird und die in der Funktion aufgerufenen lokalen Variablen enthält.)
- Filmsequenz, die das zurzeit ausgeführte Skript enthält
- Globales Objekt (vordefinierte Objekte, wie z.B. Math oder String)

In Flash 5 ersetzt die `with`-Aktion die veraltete `tellTarget`-Aktion. Die Verwendung der `with`-Aktion anstelle von `tellTarget` wird empfohlen, da es sich bei dieser Aktion um eine ActionScript-Erweiterung des ECMA-262-Standard handelt. Im Wesentlichen unterscheiden sich die `with`- und `tellTarget`-Aktionen darin, dass `with` sowohl eine Referenz auf Filmsequenzen als auch auf andere Objekte ermöglicht, wohingegen `tellTarget` einen Zielpfad zur Identifikation einer Filmsequenz erfordert und nicht auf andere Objekte verweisen kann.

Um eine Variable innerhalb einer `with`-Aktion setzen zu können, muss die Variable außerhalb der `with`-Aktion deklariert worden sein, oder Sie müssen den vollständigen Pfad zu der Zeitleiste eingeben, auf der Sie die Variable definieren möchten. Wenn Sie eine Variable innerhalb einer `with`-Aktion setzen, die nicht zuvor deklariert wurde, sucht `with` entsprechend der Reihenfolge der Gültigkeitsbereiche nach dem Wert der Variablen. Wenn die Variable noch nicht vorhanden ist, wird die neue Variable auf der Zeitleiste gesetzt, von der aus der Aufruf der `with`-Aktion erfolgt ist.

Beispiel

Im folgenden Beispiel werden die *x*- und *y*-Eigenschaften der Instanz `someOtherMovieClip` gesetzt. Anschließend wird `someOtherMovieClip` auf Bild 3 gesetzt und dort angehalten:

```
with (someOtherMovieClip) {
  _x = 50;
  _y = 100;
  gotoAndStop(3);
}
```

Der folgende Codeausschnitt hat die gleiche Funktion, verwendet jedoch keine `with`-Aktion.

```
someOtherMovieClip._x = 50;
someOtherMovieClip._y = 100;
someOtherMovieClip.gotoAndStop(3);
```

Sie können diesen Code auch unter Verwendung der `tellTarget`-Aktion formulieren.

```
tellTarget ("someOtherMovieClip") {
  _x = 50;
  _y = 100;
  gotoAndStop(3);
}
```

Die `with`-Aktion ist für den Zugriff auf mehrere Elemente in einer Reihenfolge von Gültigkeitsbereichen nützlich. Im folgenden Beispiel wird das integrierte `Math`-Objekt an der ersten Stelle der Reihenfolge der Gültigkeitsbereiche platziert. Wenn `Math` als Standardobjekt gesetzt wird, werden die Bezeichner `cos`, `sin` und `PI` in `Math.cos`, `Math.sin` und `Math.PI` aufgelöst. Die Bezeichner `a`, `x`, `y` und `r` sind keine Methoden oder Eigenschaften des `Math`-Objektes. Da sie jedoch im Objektaktivierungsbereich der Funktion `polar` vorhanden sind, werden sie in die entsprechenden lokalen Variablen aufgelöst.

```
function polar(r){
  var a, x, y
  with (Math) {
    a = PI * r * r
    x = r * cos(PI)
    y = r * sin(PI/2)
}
trace("area = " +a)
trace("x = " + x)
trace("y = " + y)
}
```

Sie können verschachtelte `with`-Aktionen für den Zugriff auf Daten in mehreren Gültigkeitsbereichen verwenden. Im folgenden Beispiel sind die Instanzen `fresno` und `salinas` untergeordnete Instanzen der Instanz `california`. Die Anweisung setzt die `_alpha`-Werte von `fresno` und `salinas`, ohne dass dabei der `_alpha`-Wert von `california` geändert wird.

```
with (california){
   with (fresno){
      _alpha = 20;
   }
   with (salinas){
      _alpha = 40;
   }
}
```

Siehe auch

„tellTarget“ auf Seite 375



_x

Syntax

instanzname._x

instanzname._x = *ganzzahl*

Argumente

ganzzahl Die lokale *x*-Koordinate des Filmes.

instanzname Der Name der Instanz einer Filmsequenz.

Beschreibung
Eigenschaft; setzt die *x*-Koordinate eines Filmes bezüglich der lokalen Koordinaten der übergeordneten Filmsequenz. Wenn sich eine Filmsequenz in der Hauptzeitleiste befindet, beziehen sich die Koordinaten auf die linke obere Ecke der Bühne (0, 0). Befindet sich eine Filmsequenz dagegen in einer anderen sich ändernden Filmsequenz, dann befindet sich die Filmsequenz im Koordinatensystem der aufnehmenden Filmsequenz. Wenn also eine Filmsequenz um 90 Grad nach links gedreht wird, dann drehen sich auch die Koordinatensysteme der untergeordneten Filme um 90 Grad nach links. Die Koordinaten einer Filmsequenz beziehen sich auf die Position des Registrierungspunktes.

Player
Flash 3 oder höher.

Siehe auch
„_y“ auf Seite 414
„_xscale“ auf Seite 413

XML (Objekt)

Verwenden Sie die Methoden und Eigenschaften des XML-Objeks zum Laden, Einlesen, Senden, Erstellen und Bearbeiten von Hierarchien von XML-Dokumenten.

Vor einem Aufruf einer Methode des XML-Objektes muss der Konstruktor `new XML()` zum Erstellen einer Instanz des XML-Objektes verwendet werden.

XML wird von Flash Player ab Version 5 unterstützt.

Zusammenfassung der Methoden für das XML-Objekt

Methode	Beschreibung
appendChild	Hängt einen Knoten am Ende der Liste der Unterelemente des angegebenen Objektes an.
cloneNode	Klont den angegebenen Knoten und optional rekursiv alle Unterelemente.
createElement	Erstellt ein neues XML-Element.
createTextNode	Erstellt einen neuen XML-Textknoten.
hasChildNodes	Gibt true zurück, wenn der angegebene Knoten über untergeordnete Knoten verfügt; andernfalls wird false zurückgegeben.
insertBefore	Fügt in der Liste der Unterelemente des angegebenen Knotens einen Knoten vor einem vorhandenen Knoten ein.
load	Lädt ein Dokument (angegeben durch das XML-Objekt) von einem URL.
onLoad	Eine Rückmeldungsfunktion für load und sendAndLoad.
parseXML	Liest ein XML-Dokument in den angegebenen Baum des XML-Objektes ein.
removeNode	Entfernt den angegebenen Knoten aus dem übergeordneten Knoten.
send	Sendet das angegebene XML-Objekt an einen URL.
sendAndLoad	Sendet das angegebene XML-Objekt an einen URL und lädt die Antwort des Servers in ein anderes XML-Objekt.
toString	Wandelt den angegebenen Knoten und alle Unterelemente in XML-Text um.

Zusammenfassung der Eigenschaften für das XML-Objekt

Eigenschaft	Beschreibung
docTypeDecl	Setzt Informationen über die DOCTYPE-Deklaration eines XML-Dokuments und gibt diese zurück.
firstChild	Verweist in der Liste für den angegebenen Knoten auf das erste Unterelement.
lastChild	Verweist in der Liste für den angegebenen Knoten auf das letzte Unterelement.
loaded	Überprüft, ob das angegebene XML-Objekt geladen wurde.

Eigenschaft	Beschreibung
`nextSibling`	Verweist auf das nächste nebengeordnete Element in der Liste der Unterelemente des übergeordneten Knotens.
`nodeName`	Gibt den Tag-Namen eines XML-Elements zurück.
`nodeType`	Gibt den Typ des angegebenen Knotens zurück (XML-Element oder Textknoten).
`nodeValue`	Gibt den Text des angegebenen Knotens zurück, wenn der Knoten ein Textknoten ist.
`parentNode`	Verweist auf den übergeordneten Knoten des angegebenen Knotens.
`previousSibling`	Verweist auf das vorige nebengeordnete Element in der Liste der Unterelemente des übergeordneten Knotens.
`status`	Gibt einen numerischen Statuscode zurück, mit dem angegeben wird, ob ein Einlesevorgang für ein XML-Dokument erfolgreich beendet wurde oder fehlgeschlagen ist.
`xmlDecl`	Setzt Informationen über die Dokumentdeklaration eines XML-Dokuments und gibt diese zurück.

Zusammenfassung der Sammlungen für das XML-Objekt

Methode	Beschreibung
`attributes`	Gibt einen assoziativen Array zurück, der alle Attribute des angegebenen Knotens enthält.
`childNodes`	Gibt einen Array mit Referenzen auf die untergeordneten Knoten des angegebenen Knotens zurück.

Konstruktor für das XML-Objekt

Syntax

```
new XML();
```

```
new XML(quelle);
```

Argumente

`quelle` Das XML-Dokument, das zum Erstellen des neuen XML-Objektes eingelesen wurde.

Beschreibung

Konstruktor; erstellt ein neues XML-Objekt. Vor einem Aufruf einer Methode des XML-Objektes muss der Konstruktor zum Erstellen einer Instanz des XML-Objektes verwendet werden.

Mit der ersten Syntax wird ein neues, leeres XML-Objekt erstellt.

Mit der zweiten Syntax wird ein neues XML-Objekt erstellt, indem das im Argument *quelle* angegebene XML-Dokument eingelesen und der so erhaltene Baum eines XML-Dokuments in das neu erstellte XML-Objekt geschrieben wird.

Anmerkung: Die `createElement`- und `createTextnode`-Methoden sind die „Konstruktormethoden" für das Erstellen der Elemente und Textknoten im Baum eines XML-Dokuments.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird ein neues, leeres XML-Objekt erstellt.

```
myXML = new XML();
```

Siehe auch

„XML.createTextNode" auf Seite 393
„XML.createElement" auf Seite 393

XML.appendChild

Syntax

meinXML.appendChild(*untergeordneterKnoten*);

Argumente

untergeordneterKnoten Der untergeordnete Knoten, der der Liste der Unterelemente des angegebenen XML-Objektes hinzugefügt werden soll.

Beschreibung

Methode; hängt den angegebenen untergeordneten Knoten an die Liste der Unterelemente des XML-Objektes an. Der angehängte untergeordnete Knoten wird in der Baumstruktur platziert, wenn er aus dem ggf. vorhandenen übergeordneten Knoten entfernt wurde.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird der letzte Knoten von `doc1` geklont und an `doc2` angehängt.

```
doc1 = new XML(src1);
doc2 = new XML();
node = doc1.lastChild.cloneNode(true);
doc2.appendChild(node);
```

XML.attributes

Syntax
meinXML.attributes;

Argumente
Keine.

Beschreibung
Sammlung (lesen/schreiben); gibt einen assoziativen Array zurück, der alle Attribute des angegebenen XML-Objektes enthält.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Beispiel werden die Namen der XML-Attribute in das Ausgabefenster geschrieben.

```
str = "<mytag name=\"Val\"> intem </mytage>";
doc = new XML(str);
y = doc.fristChild.attributes.name;
  trace (y);
doc.firstChild.attributes.order = "first";
z = doc.firstChild.attributes.order
  trace(z);
```

Folgendes wird in das Ausgabefenster geschrieben:

```
Val
First
```

XML.childNodes

Syntax
meinXML.childNodes;

Argumente
Keine.

Beschreibung
Sammlung (schreibgeschützt); gibt einen Array der Unterelemente des angegebenen XML-Objektes zurück. Jedes Element im Array ist eine Referenz auf ein XML-Objekt, das einen untergeordneten Knoten darstellt. Die Eigenschaft ist schreibgeschützt und kann nicht zum Ändern untergeordneter Knoten verwendet werden. Verwenden Sie zum Ändern von untergeordneten Knoten die `appendChild`-, `insertBefore`- und `removeNode`-Methoden.

Diese Sammlung ist für Textknoten undefiniert (nodeType == 3).

Player
Flash 5 oder höher.

XML.cloneNode

Syntax

meinXML.cloneNode(*rekursion*);

Argumente

rekursion Boolescher Wert, mit dem angegeben wird, ob die Unterelemente des angegebenen XML-Objektes rekursiv geklont werden.

Beschreibung

Methode; erstellt einen neuen XML-Knoten vom gleichen Typ, mit dem gleichen Namen, Wert und den Attributen wie das angegebene XML-Objekt und gibt diesen zurück. Wenn *rekursion* auf `true` gesetzt wurde, werden alle untergeordneten Knoten rekursiv geklont, wobei eine exakte Kopie des Dokumentbaums des Ursprungsobjektes erstellt wird.

Player

Flash 5 oder höher.

XML.createElement

Syntax

meinXML.createElement(*name*);

Argumente

name Der Tag-Name des XML-Elements, das erstellt werden soll.

Beschreibung

Methode; erstellt ein neues XML-Element mit dem Namen, der im Argument angegeben ist. Das neue Element hat zunächst kein übergeordnetes Element und keine Unterelemente. Die Methode gibt eine Referenz auf das neu erstellte XML-Objekt zurück, das das Element darstellt. Diese Methode und `createTextNode` sind die Konstruktormethoden zum Erstellen von Knoten für ein XML-Objekt.

Player

Flash 5 oder höher.

XML.createTextNode

Syntax

meinXML.createTextNode(*text*);

Argumente

text Der Text, der zum Erstellen des neuen Textknotens verwendet wird.

Beschreibung
Methode; erstellt einen neuen XML-Textknoten mit dem angegebenen Text. Das neue Element hat zunächst kein übergeordnetes Element, und Textknoten können keine Unterelemente haben. Diese Methode gibt eine Referenz auf das XML-Objekt zurück, das den neuen Textknoten darstellt. Diese Methode und `createElement` sind die Konstruktormethoden zum Erstellen von Knoten für ein XML-Objekt.

Player
Flash 5 oder höher.

XML.docTypeDecl

Syntax

```
meinXML.XMLdocTypeDecl;
```

Argumente

Keine.

Beschreibung

Eigenschaft; setzt Informationen über die DOCTYPE-Deklaration eines XML-Dokuments und gibt diese zurück. Nachdem der XML-Text in ein XML-Objekt eingelesen wurde, wird die `XML.docTypeDecl`-Eigenschaft des XML-Objektes auf den Text der DOCTYPE-Deklaration des XML-Dokuments gesetzt. Beispiel: `<!DOCTYPE greeting SYSTEM "hello.dtd">`. Diese Eigenschaft wird mit Hilfe einer Zeichenfolgendarstellung der DOCTYPE-Deklaration festgelegt, nicht mit einem XML-Knotenobjekt.

Der XML-Parser von ActionScript ist kein validierender Parser. Die DOCTYPE-Deklaration wird vom Parser gelesen und in der `docTypeDecl`-Eigenschaft gespeichert, eine Validierung der DTD wird jedoch nicht durchgeführt.

Wenn während eines Einlesevorgangs keine DOCTYPE-Deklaration gefunden wurde, wird `XML.docTypeDecl` auf undefiniert gesetzt. `XML.toString` gibt den Inhalt von `XML.docTypeDecl` unmittelbar nach der in `XML.xmlDecl` gespeicherten XML-Deklaration und vor allem sonstigen Text im XML-Objekt aus. Wenn `XML.docTypeDecl` undefiniert ist, wird keine DOCTYPE-Deklaration ausgegeben.

Player
Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird `XML.docTypeDecl` zum Setzen der DOCTYPE-Deklaration für ein XML-Objekt verwendet.

```
meinXML.docTypeDecl = "<!DOCTYPE greeting SYSTEM \"hello.dtd\">";
```

Siehe auch

„`XML.toString`“ auf Seite 404
„`XML.xmlDecl`“ auf Seite 405

XML.firstChild

Syntax

```
meinXML.firstChild;
```

Argumente

Keine.

Beschreibung

Eigenschaft (schreibgeschützt); wertet das angegebene XML-Objekt aus und verweist auf den ersten untergeordneten Knoten in der Liste der Unterelemente des übergeordneten Knotens. Diese Eigenschaft ist `null`, wenn der Knoten über keine Unterelemente verfügt. Diese Eigenschaft ist undefiniert, wenn der Knoten ein Textknoten ist. Die Eigenschaft ist schreibgeschützt und kann nicht zum Verändern untergeordneter Knoten verwendet werden. Verwenden Sie zum Verändern untergeordneter Knoten die `appendChild`-, `insertBefore`- und `removeNode`-Methoden.

Player

Flash 5 oder höher.

Siehe auch

„`XML.appendChild`“ auf Seite 391
„`XML.insertBefore`“ auf Seite 396
„`XML.removeNode`“ auf Seite 402

XML.haschildNodes

Syntax

```
meinXML.hasChildNodes();
```

Argumente

Keine.

Beschreibung

Methode; wertet das angegebene XML-Objekt aus und gibt `true` zurück, wenn untergeordnete Knoten vorhanden sind; andernfalls wird `false` zurückgegeben.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel werden die Informationen aus dem XML-Objekt in einer benutzerdefinierten Funktion verwendet.

```
if (rootNode.hasChildNodes()) {
  myfunc (rootNode.firstChild);
}
```

XML.insertBefore

Syntax

meinXML.insertBefore(*untergeordneterKnoten*, *vorKnoten*);

Argumente

untergeordneterKnoten Der einzufügende Knoten.

vorKnoten Der Knoten vor der Einfügemarke für *untergeordneterKnoten*.

Beschreibung

Methode; fügt vor dem vorKnoten einen neuen untergeordneten Knoten in die Liste der Unterelemente des XML-Objektes ein.

Player

Flash 5 oder höher.

XML.lastChild

Syntax

meinXML.lastChild;

Argumente

Keine.

Beschreibung

Eigenschaft (schreibgeschützt); wertet das angegebene XML-Objekt aus und verweist auf das letzte Unterelement in der Liste der Unterelemente des übergeordneten Knotens. Diese Methode gibt null zurück, wenn der Knoten über keine Unterelemente verfügt. Die Eigenschaft ist schreibgeschützt und kann nicht zum Verändern untergeordneter Knoten verwendet werden. Verwenden Sie zum Verändern untergeordneter Knoten die appendChild-, insertBefore- und removeNode-Methoden.

Player

Flash 5 oder höher.

Siehe auch

„XML.appendChild“ auf Seite 391
„XML.insertBefore“ auf Seite 396
„XML.removeNode“ auf Seite 402

XML.load

Syntax
meinXML.load(*URL*);

Argumente
URL Der URL, unter dem sich das zu ladende XML-Dokument befindet. Der URL muss sich in der gleichen Subdomäne befinden wie der URL, unter dem der Film momentan abgelegt ist.

Beschreibung
Methode; lädt ein XML-Dokument vom angegebenen URL und ersetzt den Inhalt des angegebenen XML-Objektes durch die heruntergeladenen XML-Daten. Der Ladevorgang ist asynchron; er wird nicht sofort nach Ausführung der `load`-Methode beendet. Wenn `load` ausgeführt wird, wird die XML-Objekteigenschaft `loaded` auf `false` gesetzt. Wenn das Herunterladen der XML-Daten abgeschlossen ist, wird die `loaded`-Eigenschaft auf `true` gesetzt und die `onLoad`-Methode aufgerufen. Die XML-Daten werden erst eingelesen, wenn sie vollständig heruntergeladen sind. Wenn das XML-Objekt bereits XML-Bäume enthielt, werden diese verworfen.

Sie können eine eigene Rückmeldungsfunktion anstelle der `onLoad`-Methode angeben.

Player
Flash 5 oder höher.

Beispiel
Im folgenden einfachen Beispiel wird die Verwendung von `XML.load` demonstriert.

```
doc = new XML();
doc.load ("theFile.xml");
```

Siehe auch
„XML.onLoad“ auf Seite 400
„XML.loaded“ auf Seite 397

XML.loaded

Syntax
meinXML.loaded;

Argumente
Keine.

Beschreibung

Eigenschaft (schreibgeschützt); gibt an, ob der durch `XML.load` initiierte Ladevorgang für das Dokument abgeschlossen ist. Wenn der Vorgang erfolgreich abgeschlossen wurde, gibt die Methode den Wert `true` zurück. Andernfalls wird der Wert `false` zurückgegeben.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird `XML.loaded` in einem einfachen Skript verwendet.

```
if (doc.loaded) {
  gotoAndPlay(4)
}
```

XML.nextSibling

Syntax

`meinXML``.nextSibling;`

Argumente

Keine.

Beschreibung

Eigenschaft (schreibgeschützt); wertet das angegebene XML-Objekt aus und verweist auf das nächste nebengeordnete Element in der Liste der Unterelemente des übergeordneten Knotens. Diese Methode gibt `null` zurück, wenn der Knoten über keine nebengeordneten Elemente verfügt. Die Eigenschaft ist schreibgeschützt und kann nicht zum Verändern untergeordneter Knoten verwendet werden. Verwenden Sie zum Ändern von untergeordneten Knoten die `appendChild`, `insertBefore`- und `removeNode`-Methoden.

Player

Flash 5 oder höher.

Siehe auch

„`XML.appendChild`“ auf Seite 391

„`XML.insertBefore`“ auf Seite 396

„`XML.removeNode`“ auf Seite 402

XML.nodeName

Syntax

`meinXML``.nodeName;`

Argumente

Keine.

Beschreibung

Eigenschaft; nimmt den Knotennamen des XML-Objektes an oder gibt diesen zurück. Wenn das XML-Objekt ein XML-Element ist (`nodeType == 1`), entspricht `nodeName` dem Namen des Tags für den Knoten in der XML-Datei. `TITLE` ist beispielsweise nodeName des HTML-Tags `TITLE`. Wenn es sich bei dem XML-Objekt um einen Textknoten handelt (`nodeType == 3`) ist nodeName `null`.

Player

Flash 5 oder höher.

Siehe auch

„`XML.nodeType`“ auf Seite 399

XML.nodeType

Syntax

`*meinXML*.nodeType;`

Argumente

Keine.

Beschreibung

Eigenschaft (schreibgeschützt); nimmt den Wert `nodeType` an oder gibt diesen zurück, wobei 1 ein XML-Element und 3 ein Textknoten ist.

Player

Flash 5 oder höher.

Siehe auch

„`XML.nodeValue`“ auf Seite 399

XML.nodeValue

Syntax

`*meinXML*.nodeValue;`

Argumente

Keine.

Beschreibung

Eigenschaft; gibt den Knotenwert des XML-Objektes zurück. Wenn es sich bei dem XML-Objekt um einen Textknoten handelt, ist nodeType 3 und `nodeValue` der Text des Knotens. Wenn das XML-Objekt ein XML-Element ist, hat `nodeValue` den Wert `null` und ist schreibgeschützt.

Player

Flash 5 oder höher.

Siehe auch

„`XML.nodeType`“ auf Seite 399

XML.onLoad

Syntax
`meinXML.onLoad(ok);`

Argumente
`ok` Ein Boolescher Wert, der anzeigt, ob das XML-Objekt mit der Operation `XML.load` oder `XML.sendAndLoad` erfolgreich geladen wurde.

Beschreibung
Methode; wird vom Flash Player aufgerufen, wenn ein XML-Dokument vom Server empfangen wird. Wenn das XML-Dokument erfolgreich empfangen wurde, hat das Argument `ok` den Wert `true`. Wenn das Dokument nicht empfangen wurde oder beim Empfang einer Antwort vom Server ein Fehler auftrat, hat das Argument `ok` den Wert `false`. Die Standardimplementierung dieser Methode ist nicht aktiv. Zum Überschreiben der Standardimplementierung müssen Sie eine Funktion mit eigenen Aktionen zuweisen.

Player
Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird ein einfacher Flash Film für eine einfache Anwendung einer E-Commerce-Storefront erstellt. Dabei wird zum Übermitteln des XML-Elements mit dem Benutzernamen und Kennwort die `sendAndLoad`-Methode verwendet und der `onLoad`-Handler installiert, um die Antwort vom Server zu verarbeiten.

```
var myLoginReply = new XML();
myLoginReply.onLoad = myOnLoad;
myXML.sendAndLoad("http://www.samplestore.com/login.cgi",
                  myLoginReply);
function myOnLoad(success) {
  if (success) {
    if (e.firstChild.nodeName == "LOGINREPLY" &&
        e.firstChild.attributes.status == "OK") {
      gotoAndPlay("loggedIn")
    } else {
      gotoAndStop("loginFailed")
    }
  } else {
    gotoAndStop("connectionFailed")
  }
}
```

Siehe auch
„`function`“ auf Seite 269
„`XML.load`“ auf Seite 397
„`XML.sendAndLoad`“ auf Seite 403

XML.parentNode

Syntax

`meinXML.parentNode;`

Argumente

Keine.

Beschreibung

Eigenschaft (schreibgeschützt); verweist auf den übergeordneten Knoten des angegebenen XML-Objektes, oder gibt den Wert `null` zurück, wenn der Knoten über keine übergeordneten Knoten verfügt. Die Eigenschaft ist schreibgeschützt und kann nicht zum Verändern untergeordneter Knoten verwendet werden. Verwenden Sie zum Verändern von Unterelementen die `appendChild`-, `insertBefore`- und `removeNode`-Methoden.

Player

Flash 5 oder höher.

XML.parseXML

Syntax

`meinXML.parseXML(quelle);`

Argumente

quelle Der XML-Text, der eingelesen und an das angegebene `XML`-Objekt übergeben werden soll.

Beschreibung

Methode; liest den im Argument *quelle* angegebenen XML-Text ein und fügt den so erhaltenen XML-Baum in das angegebene XML-Objekt ein. Alle im XML-Objekt vorhandenen Bäume werden verworfen.

Player

Flash 5 oder höher.

XML.previousSibling

Syntax

`meinXML.previousSibling;`

Argumente

Beschreibung
Eigenschaft (schreibgeschützt); wertet das angegebene XML-Objekt aus und verweist auf das vorhergehende nebengeordnete Element in der Liste der Unterelemente des übergeordneten Knotens. Es wird der Wert `null` zurückgegeben, wenn kein vorhergehender nebengeordneter Knoten vorhanden ist. Die Eigenschaft ist schreibgeschützt und kann nicht zum Verändern untergeordneter Knoten verwendet werden. Verwenden Sie zum Verändern untergeordneter Knoten die `appendChild`-, `insertBefore`- und `removeNode`-Methoden.

Player
Flash 5 oder höher.

XML.removeNode

Syntax
```
meinXML.removeNode();
```

Argumente
Keine.

Beschreibung
Methode; entfernt das angegebene XML-Objekt aus dem übergeordneten Element.

Player
Flash 5 oder höher.

XML.send

Syntax
```
meinXML.send(url);
meinXML.send(url, fenster);
```

Argumente
url Der Ziel-URL für das angegebene XML-Objekt.

fenster Das Browserfenster, in dem die vom Server zurückgesendeten Daten angezeigt werden sollen: `_self` bezeichnet den aktuellen Frame im aktuellen Fenster, `_blank` bezeichnet ein neues Fenster, `_parent` den dem aktuellen Frame übergeordneten Frame, `_top` bezeichnet den Frame der obersten Ebene des aktuellen Fensters.

Beschreibung
Methode; kodiert das angegebene XML-Objekt in ein XML-Dokument und sendet dieses mit Hilfe der `POST`-Methode an den angegebenen URL.

Player
Flash 5 oder höher.

XML.sendAndLoad

Syntax
`meinXML.sendAndLoad(url,zielXMLobjekt);`

Argumente
`url` Der Ziel-URL für das angegebene XML-Objekt. Der URL muss sich in der gleichen Subdomäne befinden wie der URL, von dem der Film heruntergeladen wurde.

`zielXMLobjekt` Ein mit der XML-Konstruktormethode erstelltes XML-Objekt, das die vom Server zurückgesendeten Informationen empfängt.

Beschreibung
Methode; kodiert das angegebene XML-Objekt in ein XML-Dokument, sendet dieses mit Hilfe der `POST`-Methode an den angegebenen URL und empfängt die Antwort des Servers und lädt diese in das in den Argumenten angegebene `zielXMLobjekt`. Die Serverantwort wird auf gleiche Weise wie mit der `load`-Methode geladen.

Player
Flash 5 oder höher.

Siehe auch
„`XML.load`“ auf Seite 397

XML.status

Syntax

`meinXML.status;`

Argumente
Keine.

Beschreibung
Eigenschaft; setzt automatisch einen numerischen Wert, der anzeigt, ob das XML-Dokument erfolgreich in ein XML-Objekt eingelesen wurde, und gibt diesen zurück. In der folgenden Liste werden die einzelnen numerischen Statuscodes mit einer Beschreibung aufgeführt.

- 0 Kein Fehler; Einlesevorgang erfolgreich abgeschlossen.
- -2 Ein CDATA-Bereich wurde nicht ordnungsgemäß beendet.
- -3 Die XML-Deklaration wurde nicht ordnungsgemäß beendet.
- -4 Die DOCTYPE-Deklaration wurde nicht ordnungsgemäß beendet.

- -5 Ein Kommentar wurde nicht ordnungsgemäß beendet.
- -6 Ein XML-Element war ungültig.
- -7 Zu wenig Speicher.
- -8 Ein Attributwert wurde nicht ordnungsgemäß beendet.
- -9 Der zu einem Anfangstag gehörige Endtag fehlte.
- -10 Es trat ein Endtag ohne zugehörigen Anfangstag auf.

Player
Flash 5 oder höher.

XML.toString

Syntax
`meinXML.toString();`

Argumente
Keine.

Beschreibung
Methode; wertet das angegebene XML-Objekt aus, erstellt eine Textdarstellung der XML-Struktur mit Knoten, Unterelementen und Attributen und gibt das Ergebnis als Zeichenfolge zurück.

Bei XML-Objekten der höchsten Ebene (mit dem Konstruktor erstellt) gibt `XML.toString` die XML-Deklaration des Dokuments aus (gespeichert in `XML.xmlDecl`), gefolgt von der `DOCTYPE`-Deklaration (gespeichert in `XML.docTypeDecl`) und der Textdarstellung aller XML-Knoten im Objekt. Es wird keine XML-Deklaration ausgegeben, wenn `XML.xmlDecl` undefiniert ist. Die `DOCTYPE`-Deklaration wird nicht ausgegeben, wenn `XML.docTypeDecl` undefiniert ist.

Player
Flash 5 oder höher.

Beispiel
Im folgenden Codebeispiel wird die Verwendung der `XML.toString`-Methode demonstriert.

```
node = new XML("<h1>test</h1>");
trace(node.toString());
sendet
<H1>test</H1>
an das Ausgabefenster
```

Siehe auch
„XML.xmlDecl“ auf Seite 405
„XML.docTypeDecl“ auf Seite 394

XML.xmlDecl

Syntax

meinXML.xmlDecl;

Argumente

Keine.

Beschreibung

Eigenschaft; setzt die Informationen für die XML-Deklaration eines Dokuments und gibt diese zurück. Nachdem das XML-Dokument in ein XML-Objekt eingelesen wurde, wird diese Eigenschaft entsprechend dem Text der XML-Deklaration des Dokuments gesetzt. Diese Eigenschaft wird mit einer Darstellung der XML-Deklaration als Zeichenfolge und nicht eines XML-Knotenobjektes gesetzt. Wenn während eines Einlesevorgangs keine XML-Deklaration gefunden wurde, wird diese Eigenschaft auf undefiniert gesetzt. `XML.toString` gibt den Inhalt von `XML.xmlDecl` vor anderem Text im XML-Objekt aus. Wenn `XML.xmlDecl` den Typ `undefined` enthält, wird keine XML-Deklaration ausgegeben.

Player

Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird `XML.xmlDecl` zum Setzen der XML-Deklaration des Dokuments für ein XML-Objekt verwendet.

```
myXML.xmlDecl = "<?xml version=\"1.0\" ?>";
```

Siehe auch

„XML.toString“ auf Seite 404
„XML.docTypeDecl“ auf Seite 394

XMLSocket (Objekt)

Das XMLSocket-Objekt implementiert Clientsockets, mit denen der Computer über Flash Player mit einem Servercomputer kommunizieren kann, der durch eine IP-Adresse oder einen Domänennamen festgelegt ist.

Verwenden des XMLSocket-Objekts

Zum Verwenden des XMLSocket-Objektes muss auf dem Servercomputer ein Daemon ausgeführt werden, der das vom XMLSocket-Objekt verwendete Protokoll verarbeiten kann. Das Protokoll lautet wie folgt:

- XML-Nachrichten werden über eine TCP/IP-Streaming-Socket-Verbindung im Vollduplexmodus gesendet.
- Jede XML-Nachricht ist ein vollständiges XML-Dokument, das mit einem Null-Byte abgeschlossen wird.

- Es kann eine unbegrenzte Anzahl von XML-Nachrichten über eine einzelne XMLSocket-Verbindung gesendet und empfangen werden.

Das XMLSocket-Objekt ist für Client-Server-Anwendungen nützlich, die geringe Latenzzeiten verlangen, beispielsweise Echtzeit-Chat-Systeme. Eine herkömmliche HTTP-Chat-Lösung fragt den Server in kurzen Abständen ab und lädt neue Nachrichten mit Hilfe einer HTTP-Anforderung. Im Gegensatz dazu unterhält eine XMLSocket-Chat-Lösung eine offene Verbindung zum Server, wodurch dieser neu eingegangene Nachrichten sofort und ohne Anforderung vom Client senden kann.

Das Einrichten eines Servers für die Kommunikation mit einem XMLSocket-Objekt ist möglicherweise mit Schwierigkeiten verbunden. Wenn Ihre Anwendung keine Echtzeitinteraktion erforderlich macht, sollten Sie die `loadVariables`-Aktion oder die HTTP-XML-Serververbindung von Flash (`XML.load`, `XML.sendAndLoad`, `XML.send`) anstelle des XMLSocket-Objektes verwenden.

Bevor Sie Methoden des XMLSocket-Objektes verwenden können, müssen Sie zuerst den Konstruktor `new XMLSocket` verwenden, um ein neues XMLSocket-Objekt zu erstellen.

XMLSocket und Sicherheit

Da das XMLSocket-Objekt eine offene Verbindung zum Server herstellt und verwaltet, wurden aus Sicherheitsgründen die folgenden Einschränkungen im XML-Socket-Objekt implementiert.

- Die `XMLSocket.connect`-Methode kann nur Verbindungen zu TCP-Portnummern herstellen, die größer oder gleich 1024 sind. Als Folge dieser Einschränkung müssen den Serverdaemons, die mit dem XMLSocket-Objekt kommunizieren, auch Portnummern zugeordnet sein, die größer oder gleich 1024 sind. Portnummern unter 1024 werden häufig von Systemdiensten wie FTP, Telnet und HTTP verwendet und blockieren so die Verwendung dieser Anschlüsse durch das XMLSocket-Objekt. Die Einschränkung für Portnummern begrenzt die Möglichkeit, dass in ungeeigneter oder missbräuchlicher Weise auf diese Ressourcen zugegriffen wird.
- Mit der `XMLSocket.connect`-Methode können nur Verbindungen zu Computern hergestellt werden, die sich in der gleichen Subdomäne wie die SWF-Datei (Flash Film) befinden. Diese Einschränkung gilt nicht für Flash Filme, die von einem lokalen Datenträger wiedergegeben werden. (Diese Einschränkung ist mit den Sicherheitsvorschriften für `loadVariables`, `XML.sendAndLoad` und `XML.load` identisch.)

Zusammenfassung der Methoden für das XMLSocket-Objekt

Methode	Beschreibung
`close`	Beendet eine offene Socket-Verbindung.
`connect`	Stellt eine Verbindung zum angegebenen Server her.
`onClose`	Eine Rückmeldungsfunktion, die beim Beenden einer XMLSocket-Verbindung aufgerufen wird.
`onConnect`	Eine Rückmeldungsfunktion, die beim Herstellen einer XMLSocket-Verbindung aufgerufen wird.
`onXML`	Eine Rückmeldungsfunktion, die beim Empfangen eines XML-Objektes vom Server aufgerufen wird.
`send`	Sendet ein XML-Objekt an den Server.

Konstruktor für das XMLSocket-Objekt

Syntax

```
new XMLSocket();
```

Argumente

Keine.

Beschreibung

Konstruktor; erstellt ein neues XMLSocket-Objekt. Das XMLSocket-Objekt ist zunächst nicht mit einem Server verbunden. Sie müssen Sie die `XMLSocket.connect`-Methode aufrufen, um das Objekt mit einem Server zu verbinden.

Player

Flash 5 oder höher.

Beispiel

```
myXMLSocket = new XMLSocket();
```

Siehe auch

„XMLSocket.connect“ auf Seite 408

XMLSocket.close

Syntax

```
meinXMLSocket.close();
```

Argumente

Keine.

Beschreibung

Methode; beendet die durch das XMLSocket-Objekt angegebene Verbindung.

Player
Flash 5 oder höher.

Siehe auch
„XMLSocket.connect“ auf Seite 408

XMLSocket.connect

Syntax
`meinXMLSocket.connect(host, port);`

Argumente
`host` Ein vollqualifizierter DNS-Domänenname oder eine IP-Adresse der Form *aaa.bbb.ccc.ddd.* Sie können auch `null` angeben, um eine Verbindung mit dem Hostserver herzustellen, auf dem der Film gespeichert ist.

`port` Die zur Herstellung einer Verbindung verwendete TCP-Portnummer auf dem Host. Die Portnummer muss 1024 oder höher sein.

Beschreibung
Methode; stellt eine Verbindung mit dem angegebenen Internethost unter dem angegebenen TCP-Port (1024 oder höher) her und gibt bei erfolgreichem Verbindungsaufbau den Wert `true` zurück; andernfalls den Wert `false`. Wenn Sie die Portnummer des Internet-Hostcomputers nicht kennen, wenden Sie sich an Ihren Netzwerkadministrator. Wenn das Flash Plug-In für Netscape oder das ActiveX-Steuerelement verwendet wird, muss der im Argument angegebene Host sich in der gleichen Subdomäne wie der Host befinden, von dem der Film heruntergeladen wurde.

Wenn Sie für das Argument `host` den Wert `null` angeben, wird der Host kontaktiert, auf dem der Film gespeichert ist, der `XMLSocket.connect` aufgerufen hat. Wenn der Film beispielsweise von http://www.ihresite.de heruntergeladen wurde, entspricht die Angabe von `null` für das Argument `host` der Eingabe der IP-Adresse für www.ihresite.de.

Wenn `XMLSocket.connect` den Wert `true` zurückgibt, ist der erste Schritt des Verbindungsaufbaus erfolgreich. Zu einem späteren Zeitpunkt wird die `XMLSocket.onConnect`-Methode gestartet, um festzustellen, ob die endgültige Verbindung erfolgreich hergestellt werden konnte. Wenn `XMLSocket.connect` den Wert `false` zurückgibt, konnte keine Verbindung hergestellt werden.

Player
Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird mit `XMLSocket.connect` eine Verbindung mit dem Host hergestellt, auf dem der Film gespeichert ist, wobei mit `trace` der Rückgabewert angezeigt wird, der Auskunft darüber gibt, ob die Verbindung hergestellt werden konnte.

```
function myOnConnect(success) {
  if (success) {
    trace ("Verbindungsaufbau erfolgreich.")
  } else {
    trace ("Verbindungsaufbau fehlgeschlagen.")
  }
}
socket = new XMLSocket()
socket.onConnect = myOnConnect
if (!socket.connect(null, 2000)) {
  trace ("Verbindungsaufbau fehlgeschlagen.")
}
```

Siehe auch

„`function`“ auf Seite 269
„`XMLSocket.onConnect`“ auf Seite 410

XMLSocket.onClose

Syntax

`meinXMLSocket``.onClose();`

Argumente

Keine.

Beschreibung

Methode; eine Rückmeldungsfunktion, die aufgerufen wird, wenn eine offene Verbindung vom Server getrennt wird. Die Standardimplementierung dieser Methode führt keine Aktionen aus. Zum Überschreiben der Standardimplementierung müssen Sie eine Funktion mit eigenen Aktionen zuweisen.

Player

Flash 5 oder höher.

Siehe auch

„`function`“ auf Seite 269
„`XMLSocket.onConnect`“ auf Seite 410

XMLSocket.onConnect

Syntax

meinXMLSocket.onConnect(*ok*);

Argumente

ok Ein Boolescher Wert, der anzeigt, ob eine Socket-Verbindung erfolgreich hergestellt wurde (`true` oder `false`).

Beschreibung

Methode; eine Rückmeldungsfunktion, die von Flash Player aufgerufen wird, wenn die Verbindungsanforderung durch die `XMLSocket.connect`-Methode erfolgreich war oder fehlgeschlagen ist. Wenn die Verbindung erfolgreich hergestellt wurde, hat das Argument *ok* den Wert `true`, andernfalls den Wert `false`.

Die Standardimplementierung dieser Methode führt keine Aktionen aus. Zum Überschreiben der Standardimplementierung müssen Sie eine Funktion mit eigenen Aktionen zuweisen.

Player

Flash 5 oder höher.

Beispiel

Das folgende Beispiel demonstriert die Definition einer Ersatzfunktion für die `onConnect`-Methode in einer einfachen Chat-Anwendung.

Die Funktion steuert, welcher Bildschirm dem Benutzer jeweils angezeigt wird, wenn eine Verbindung erfolgreich hergestellt werden konnte oder nicht. Wenn die Verbindung erfolgreich hergestellt wurde, wird dem Benutzer im Bild mit der Bezeichnung `startChat` der Hauptbildschirm für den Chat angezeigt. Wenn keine Verbindung hergestellt werden konnte, wird dem Benutzer im Bild mit der Bezeichnung `connectionFailed` ein Bildschirm mit Hilfestellungen bei der Problemlösung angezeigt.

```
function myOnConnect(success) {
  if (success) {
    gotoAndPlay("startChat")
  } else {
    gotoAndStop("connectionFailed")
  }
}
```

Nach dem Erstellen des XMLSocket-Objektes mit der Konstruktormethode installiert das Skript die `onConnect`-Methode mit dem Zuweisungsoperator:

```
socket = new XMLSocket()
socket.onConnect = myOnConnect
```

Schließlich wird die Verbindung initiiert. Wenn `connect` den Wert `false` zurückgibt, springt der Film direkt an das Bild mit der Bezeichnung `connectionFailed`, und `onConnect` wird nicht gestartet. Wenn `connect` den Wert `true` zurückgibt, springt der Film zu dem Bild mit der Bezeichnung `waitForConnection`, dies ist der Bildschirm „Bitte warten“. Der Film bleibt bei dem Bild `waitForConnection`, bis der `onConnect`-Handler aufgerufen wurde. Dies kann einige Zeit dauern, je nach Latenzzeit des Netzwerks.

```
if (!socket.connect(null, 2000)) {
  gotoAndStop("connectionFailed")
} else {
  gotoAndStop("waitForConnection")
}
```

Siehe auch

„XMLSocket.connect“ auf Seite 408
„function“ auf Seite 269

XMLSocket.onXML

Syntax

meinXMLSocket.onXML(*objekt*);

Argument

objekt Eine Instanz des XML-Objektes mit einem eingelesenen XML-Dokument, das von einem Server empfangen wurde.

Beschreibung

Methode; eine Rückmeldungsfunktion, die von Flash Player aufgerufen wird, wenn das angegebene XML-Objekt mit einem XML-Dokument über eine offene XMLSocket-Verbindung empfangen wird. Über eine XMLSocket-Verbindung kann eine unbegrenzte Anzahl von XML-Dokumenten zwischen dem Client und dem Server übermittelt werden. Jedes Dokument wird durch ein Null-Byte abgeschlossen. Wenn der Flash Player das Null-Byte empfängt, werden alle seit dem letzten Null-Byte empfangenen XML-Daten eingelesen. Wenn dies die erste empfangene Nachricht ist, werden alle seit Verbindungsaufbau empfangenen XML-Daten eingelesen. Jeder Abschnitt eingelesener XML-Daten wird wie ein einzelnes XML-Dokument behandelt und an die `onXML`-Methode übergeben.

Die Standardimplementierung dieser Methode führt keine Aktionen aus. Zum Überschreiben der Standardimplementierung müssen Sie eine Funktion mit eigenen Aktionen zuweisen.

Player

Flash 5 oder höher.

Beispiel
Die folgende Funktion überschreibt die Standardimplementierung der `onXML`-Methode in einer einfachen Chat-Anwendung. Die Funktion `myOnXML` weist die Chat-Anwendung an, das einzelne XML-Element `MESSAGE` in folgendem Format zu erkennen:

```
<MESSAGE USER="Jochen" TEXT="Hallo, mein Name ist Jochen!" />.
```

Der `onXML`-Handler muss zuerst mit dem folgenden Verfahren im XMLSocket-Objekt installiert werden:

```
socket.onXML = myOnXML;
```

Die Funktion `displayMessage` wird als eine benutzerdefinierte Funktion angesehen, die dem Benutzer die empfangene Nachricht anzeigt.

```
function myOnXML(doc) {
  var e = doc.firstChild;
  if (e != null && e.nodeName == "MESSAGE") {
    displayMessage(e.attributes.user, e.attributes.text);
  }
}
```

Siehe auch
„`function`“ auf Seite 269

XMLSocket.send

Syntax
`meinXMLSocket``.send(`*`objekt`*`);`

Argumente
`objekt` Ein XML-Objekt oder andere Daten für die Übermittlung an den Server.

Beschreibung
Methode; wandelt das im Argument *`objekt`* angegebene XML-Objekt oder die Daten in eine Zeichenfolge um und übermittelt diese gefolgt von einem Null-Byte an den Server. Wenn es sich bei *`objekt`* um ein XML-Objekt handelt, ist die Zeichenfolge eine Darstellung in XML-Text des XML-Objektes. Die send-Operation ist asynchron; sie antwortet unmittelbar, die Daten werden aber möglicherweise zu einem späteren Zeitpunkt gesendet. Die `XMLSocket.send`-Methode gibt keinen Wert zurück, der angibt, ob die Daten erfolgreich übermittelt wurden.

Wenn das *`meinXMLSocket`*-Objekt über keine Verbindung zu dem Server verfügt (mit Hilfe von `XMLSocket.connect`), schlägt die `XMLSocket.send`-Operation fehl.

Player
Flash 5 oder höher.

Beispiel

Im folgenden Beispiel wird demonstriert, wie Sie einen Benutzernamen und ein Kennwort zum Senden des XML-Objektes `myXML` an den Server festlegen.

```
var myXML = new XML();
var myLogin = myXML.createElement("login");
myLogin.attributes.username = usernameTextField;
myLogin.attributes.password = passwordTextField;
myXML.appendChild(myLogin);
myXMLSocket.send(myXML);
```

Siehe auch

„XMLSocket.connect“ auf Seite 408

_xmouse

Syntax

instanzname._xmouse

Argumente

instanzname Der Name der Instanz einer Filmsequenz.

Beschreibung

Eigenschaft (schreibgeschützt); sendet die *x*-Koordinate der Mausposition zurück.

Player

Flash 5 oder höher.

Siehe auch

„Mouse (Objekt)“ auf Seite 308
„_ymouse“ auf Seite 414

_xscale

Syntax

instanzname._xscale

instanzname._xscale = *prozent*;

Argumente

prozent Ein Prozentwert, der die horizontale Skalierung des Filmes angibt. Der Standardwert ist 100.

instanzname Der Name der Instanz einer Filmsequenz.

Beschreibung

Eigenschaft; legt die horizontale Skalierung (*prozent*) der Filmsequenz fest, die ausgehend vom Registrierpunkt einer Filmsequenz angelegt wird. Der Standardfeststellpunkt ist (0,0).

Das Skalieren des lokalen Koordinatensystems beeinflusst die Einstellungen für die Eigenschaften `_x` und `_y`, die in Pixel angegeben werden. Wenn die übergeordnete Filmsequenz auf 50 % skaliert ist, verschiebt das Setzen der `_x`-Eigenschaft ein Objekt in der Filmsequenz um die halbe Anzahl von Pixel, die bei 100 % zugewiesen würden.

Player
Flash 4 oder höher.

Siehe auch
„`_xscale`“ auf Seite 413



_y

Syntax

```
instanzname._y
instanzname._y = ganzzahl;
```

Argumente

`ganzzahl` Die lokale *y*-Koordinate der Filmsequenz.

`instanzname` Der Name der Instanz einer Filmsequenz.

Beschreibung
Eigenschaft; setzt die *y*-Koordinate des Filmes relativ zu den lokalen Koordinaten der übergeordneten Filmsequenz. Wenn sich eine Filmsequenz in der Hauptzeitleiste befindet, bezieht sich das Koordinatensystem auf die linke obere Ecke der Bühne (0,0). Befindet sich eine Filmsequenz dagegen in einer anderen sich ändernden Filmsequenz, dann befindet sich die Filmsequenz im Koordinatensystem der aufnehmenden Filmsequenz. Wenn also eine Filmsequenz um 90 Grad nach links gedreht wird, dann drehen sich auch die Koordinatensysteme der untergeordneten Filme um 90 Grad nach links. Die Koordinaten einer Filmsequenz beziehen sich auf die Position des Registrierungspunktes.

Player
Flash 3 oder höher.

Siehe auch
„`_yscale`“ auf Seite 415

_ymouse

Syntax

```
instanzname._ymouse
```

Argumente

`instanzname` Der Name der Instanz einer Filmsequenz.

Beschreibung
Eigenschaft (schreibgeschützt); gibt die *y*-Koordinate der Mausposition an.

Player
Flash 5 oder höher.

Siehe auch
„Mouse (Objekt)“ auf Seite 308
„_xmouse“ auf Seite 413

_yscale

Syntax
```
instanzname._yscale
instanzname._yscale = prozent;
```

Argumente
prozent Ein Prozentwert, der die vertikale Skalierung des Filmes angibt. Der Standardwert ist 100.

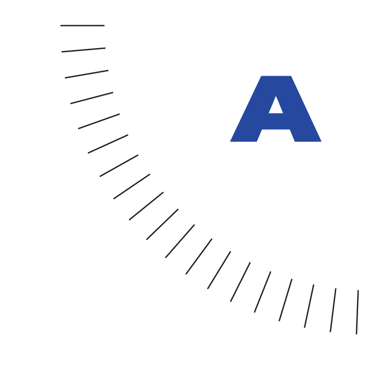
instanzname Der Name der Instanz einer Filmsequenz.

Beschreibung
Eigenschaft; legt die vertikale Skalierung (*prozent*) der Filmsequenz fest, die ausgehend vom Registrierpunkt einer Filmsequenz angelegt wird. Der Standardregistrierpunkt ist (0,0).

Das Skalieren des lokalen Koordinatensystems beeinflusst die Einstellungen für die Eigenschaften `_x` und `_y`, die in Pixel angegeben werden. Wenn die übergeordnete Filmsequenz auf 50 % skaliert ist, verschiebt das Setzen der `_x`-Eigenschaft ein Objekt in der Filmsequenz um die halbe Anzahl von Pixel, die bei 100 % zugewiesen würden.

Player
Flash 4 oder höher.

Siehe auch
„_x“ auf Seite 387
„_y“ auf Seite 414



ANHANG A
Vorrang und Assoziativität von Operatoren

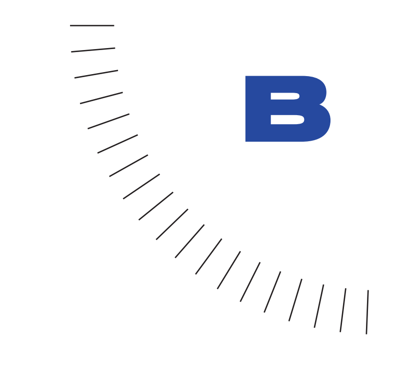
Operatorenliste

In dieser Tabelle werden alle Operatoren von ActionScript und deren Assoziativität von höchstem bis niedrigstem Vorrang aufgeführt.

Operator	Beschreibung	Assoziativität
	Höchster Vorrang	
+	Unäres Plus	Von rechts nach links
-	Unäres Minus	Von rechts nach links
~	Bitweise Negation (Bit-Komplement)	Von rechts nach links
!	Logisches NICHT	Von rechts nach links
not	Logisches NICHT (wie in Flash 4)	Von rechts nach links
++	Post-Inkrement	Von links nach rechts
--	Post-Dekrement	Von links nach rechts
( )	Funktionsaufruf	Von links nach rechts
[ ]	Array-Element	Von links nach rechts
.	Strukturelement	Von links nach rechts
++	Prä-Inkrement	Von rechts nach links
--	Prä-Dekrement	Von rechts nach links
new	Objekt zuweisen	Von rechts nach links

Operator	Beschreibung	Assoziativität
delete	Objektzuweisung aufheben	Von rechts nach links
typeof	Typ des Objekts	Von rechts nach links
void	Gibt einen undefinierten Wert zurück	Von rechts nach links
*	Multiplizieren	Von links nach rechts
/	Dividieren	Von links nach rechts
%	Modulo	Von links nach rechts
+	Addieren	Von links nach rechts
add	Zeichenfolgenverbindung (früher &)	Von links nach rechts
-	Subtrahieren	Von links nach rechts
<<	Bitweises Shift links	Von links nach rechts
>>	Bitweises Shift rechts	Von links nach rechts
>>>	Bitweises Shift rechts (ohne Vorzeichen)	Von links nach rechts
<	Kleiner als	Von links nach rechts
<=	Kleiner oder gleich	Von links nach rechts
>	Größer als	Von links nach rechts
>=	Größer oder gleich	Von links nach rechts
lt	Kleiner als (für Zeichenfolgen)	Von links nach rechts
le	Kleiner oder gleich (für Zeichenfolgen)	Von links nach rechts
gt	Größer als (für Zeichenfolgen)	Von links nach rechts
ge	Größer oder gleich (für Zeichenfolgen)	Von links nach rechts
==	Gleich	Von links nach rechts
!=	Ungleich	Von links nach rechts
eq	Gleich (für Zeichenfolgen)	Von links nach rechts
ne	Ungleich (für Zeichenfolgen)	Von links nach rechts
&	Bitweises UND	Von links nach rechts
^	Bitweises XODER	Von links nach rechts
|	Bitweises ODER	Von links nach rechts
&&	Logisches UND	Von links nach rechts

<table>
<tr><th>Operator</th><th>Beschreibung</th><th>Assoziativität</th></tr>
<tr><td>and</td><td>Logisches UND (Flash 4)</td><td>Von links nach rechts</td></tr>
<tr><td>||</td><td>Logisches ODER</td><td>Von links nach rechts</td></tr>
<tr><td>or</td><td>Logisches ODER (Flash 4)</td><td>Von links nach rechts</td></tr>
<tr><td>?:</td><td>Bedingt</td><td>Von rechts nach links</td></tr>
<tr><td>=</td><td>Zuweisung</td><td>Von rechts nach links</td></tr>
<tr><td>„*=, /=, %=, +=, -=, &=, |=, ^=, ‹‹=, ››=, ›››=“</td><td>Zusammengesetzte Zuweisung</td><td>Von rechts nach links</td></tr>
<tr><td>,</td><td>Mehrfache Auswertung</td><td>Von links nach rechts</td></tr>
<tr><td colspan="3">Niedrigster Vorrang</td></tr>
</table>



ANHANG B
Tastatureingaben und Tastencodewerte

In den folgenden Tabellen werden alle Tasten auf einer Standardtastatur und die entsprechenden Tastencodewerte aufgeführt, mit denen diese Tasten in ActionScript identifiziert werden können. Weitere Informationen finden Sie in der Beschreibung des Key-Objektes in Kapitel 7, „Referenz zu ActionScript.“

Buchstaben A bis Z und Ziffern 0 bis 9

Buchstaben- oder Zahlentaste	Tastencode
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
0	48

Buchstaben- oder Zahlentaste	Tastencode
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Tasten auf dem numerischen Ziffernblock

Taste auf dem numerischen Ziffern-block	Tastencode
0	96
1	97
2	98
3	99
4	100
5	101
6	102
7	103
8	104
9	105
MULTIPLIZIEREN	106
PLUS	107
EINGABE	108
MINUS	109
KOMMA	110
DIVIDIEREN	111

Funktionstasten

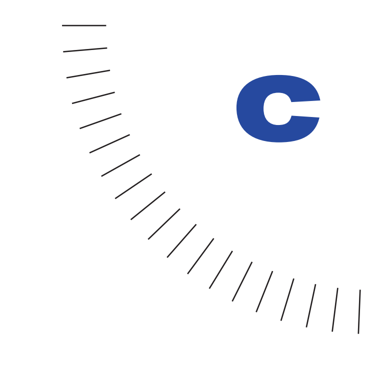
Funktionstaste	Tastencode
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117

Funktionstaste	Tastencode
F7	118
F8	119
F9	120
F10	121
F11	122
F12	123

Andere Tasten

Taste	Tastencode
RÜCKTASTE	8
TAB	9
ENTF	12
EINGABE	13
Umschalt	16
STRG	17
Alt	18
FESTSTELLTASTE	20
Esc	27
LEERTASTE	32
BILD-AUF	33
BILD-AB	34
ENDE	35
POS1	36
NACH-LINKS	37
NACH-OBEN	38
NACH-RECHTS	39
NACH-UNTEN	40
EINFG	45
RÜCKSCHRITT	46
HILFE	47
NUM	144
; :	186
= +	187
- _	189
/ ?	191
` ~	192

Taste	Tastencode
[{	219
\ |	220
] }	221
“ ‘	222



ANHANG C
Fehlermeldungen

Die folgende Tabelle enthält eine Liste von Fehlermeldungen, die vom Flash Compiler ausgegeben werden. Zu jeder Meldung wird eine Erklärung gegeben, damit Sie die Probleme in Ihren Filmdateien leichter beheben können.

Fehlermeldung	Beschreibung
Eigenschaft ‹property› existiert nicht	Es wurde eine nicht vorhandene Eigenschaft gefunden. Beispiel: `x = _green` ist ungültig, weil die Eigenschaft `_green` nicht vorhanden ist.
Nach Operator ‹operator› muss ein Operand folgen.	Es wurde ein Operator ohne Operand gefunden. Beispiel: In `x = 1 +` ist nach dem Operator + ein Operand erforderlich. Nach einem Operator steht ein ungültiger Operand. Beispiel: `trace(1+);` ist syntaktisch fehlerhaft.
Syntaxfehler	Diese Meldung wird ausgegeben, wenn ein nicht weiter spezifierter Syntaxfehler gefunden wird.
Feldname hinter '.'-Operator erwartet.	Sie müssen einen gültigen Feldnamen eingeben, wenn Sie die Syntax *`objekt.feld`* verwenden.

Expected ‹token›	Ein ungültiges oder nicht erwartetes Token wurde gefunden. Beispiel: In der folgenden Syntax ist das Token `foo` ungültig. Es wird das Token `while` erwartet.
	`do {`
	`    trace (i)`
	`  } foo (i < 100)`

Fehlermeldung	Beschreibung
Initialisierliste muss mit ‹terminator› abgeschlossen werden	In einer Liste für Objekt- oder Arrayinitialisierliste fehlt der Abschluss durch] oder }.
Bezeichner erwartet	Anstelle eines Bezeichners wurde ein unerwartetes Token gefunden. Im folgenden Beispiel ist 3 kein gültiger Bezeichner. `var 3 = 4;`
Das JavaScript-Konstrukt ‹construct› wird nicht unterstützt	Es wurde ein JavaScript-Konstrukt gefunden, das von ActionScript nicht unterstützt wird. Diese Nachricht wird angezeigt, wenn folgende JavaScript-Konstrukte verwendet werden: `void`, `switch`, `try`, `catch` oder `throw`.
Die linke Seite eines Zuweisungsoperators muss eine Variable oder eine Eigenschaft sein	Es wurde ein Zuweisungsoperator gefunden. Auf der linken Seite der Zuweisung steht jedoch weder eine gültige Variable noch eine Eigenschaft.
Anweisungsblock muss mit '}' abgeschlossen werden	Innerhalb von geschweiften Klammern wurde eine Gruppe von Anweisungen deklariert, aber die abschließende Klammer fehlt.
Ereignis erwartet	Es wurde ein Handler `On(MouseEvent)` oder `onClipEvent` deklariert, aber kein Ereignis angegeben, oder an der für das Ereignis vorgesehenen Stelle wurde ein unerwartetes Token gefunden.
Nicht zulässiges Ereignis	Das Skript enthält ein ungültiges Maus- oder Sequenzereignis. Eine Liste der gültigen Maus- und Sequenzereignisse finden Sie unter den Einträgen „On(MouseEvent)“ und „OnClipEvent“ im Kapitel „Referenz zu ActionScript“.
Tastencode erwartet	Sie müssen einen Tastencode angeben. Eine Liste der Tastencodes finden Sie in Anhang B.
Nicht zulässiger Tastencode	Der angegebene Tastencode existiert nicht.
Nicht notwendige Daten als Anhang gefunden	Das Skript oder der Ausdruck wurde ordnungsgemäß ausgewertet. Die darauf folgenden Zeichen konnten jedoch nicht geparst werden.

Fehlermeldung	Beschreibung
Unzulässige Funktion	Eine Deklaration einer benannten Funktion wurde als Ausdruck verwendet. Deklarationen von benannten Funktionen müssen Anweisungen sein. Gültig: `function sqr (x) { return x * x; }` Ungültig: `var v = function sqr (x) { return x * x; }`
Funktionsname erwartet	Der für diese Funktion angegebene Name ist kein gültiger Funktionsname.
Parametername erwartet	In einer Funktionsdeklaration wurde ein Name für einen Parameter (Argument) erwartet. Es wurde aber ein unerwartetes Token gefunden.
'else' ohne entsprechendes 'if' festgestellt	Es wurde die Anweisung `else` gefunden. Dieser geht jedoch kein `if` voran. Sie können `else` nur in Verbindung mit einer `if` -Anweisung verwenden.
Szenenname muss eine Zeichenfolge in Anführungszeichen sein	Das Szenenargument einer Aktion `gotoAndPlay`, `gotoAndStop` oder `ifFrameLoaded` ist vom falschen Typ. Das Szenenargument muss eine Zeichenfolgenkonstante sein.
Interner Fehler	Im ActionScript-Compiler ist ein interner Fehler aufgetreten. Senden Sie die FLA-Datei, bei der dieser Fehler aufgetreten ist, und detaillierte Anweisungen, wie die Meldung reproduziert werden kann, an Macromedia.
Hexadezimale Zeichen nach 0x erwartet	Es wurde die Zeichenfolge 0x gefunden. Auf diese Zeichenfolge folgt jedoch keine gültige Hexadezimalzahl.
Fehler beim Öffnen von include-Datei [#include file]	Beim Öffnen einer mit der Direktive `include` eingebundenen Datei ist ein Fehler aufgetreten. Dieser Fehler tritt z.B. auf, wenn die Datei nicht vorhanden ist oder ein Festplattenfehler vorliegt.
Ungültige #include-Anweisung	Eine `include`-Direktive wurde fehlerhaft geschrieben. Eine `include`-Direktive muss die folgende Syntax aufweisen: `#include "`*`einedatei`*`.as"`
Mehrzeiliger Kommentar wurde nicht abgeschlossen	Bei einem mehrzeiligen Kommentar, der mit `/*` beginnt, fehlen die Zeichen `*/` als Abschluss.

Fehlermeldung	Beschreibung
Zeichenfolgenliteral wurde nicht ordnungsgemäß abgeschlossen	Bei einem Zeichenfolgenliteral, das mit einem einfachen oder doppelten Anführungszeichen beginnt, fehlt das schließende Anführungszeichen.
Funktion ‹function› hat ‹count› Parameter	Eine Funktion wurde aufgerufen. Dabei wurde eine unerwartete Anzahl an Parametern gefunden.
Eigenschaftenname in GetProperty erwartet	Die Funktion `getProperty` wurde aufgerufen. Beim zweiten Argument handelt es sich jedoch nicht um den Namen einer Filmsequenzeigenschaft.
Parameter ‹parameter› darf nicht mehrfach deklariert werden	In der Parameterliste einer Funktionsdeklaration ist ein Parametername mehrmals aufgeführt. Alle Parameternamen müssen eindeutig sein.
Variable ‹variable› darf nicht mehrfach deklariert werden	In einer `var`-Anweisung ist ein Variablenname mehrfach aufgeführt. In einer einzelnen `var`-Anweisung müssen alle Variablennamen eindeutig sein.
‘on’-Handler dürfen nicht in andere ‘on’ verschachtelt werden	Ein `on`-Handler wurde ein innerhalb eines anderen `on`-Handlers deklariert. Alle `on`-Handler müssen sich auf der obersten Ebene einer Aktionsliste befinden.
Anweisung muss innerhalb eines [on]-Handlers stehen	In den Aktionen für eine Schaltflächeninstanz wurde eine Anweisung ohne einen umgebenden `on`-Block deklariert. Alle Aktionen für eine Schaltflächeninstanz müssen sich in einem `on`-Block befinden.
Anweisung muss innerhalb eines onClipEvent-Handlers stehen	In den Aktionen für eine Filmsequenzinstanz wurde eine Anweisung ohne einen umgebenden `onClipEvent`-Block deklariert. Alle Aktionen für eine Filmsequenzinstanz müssen sich in einem `onClipEvent`-Block befinden.
Mausereignisse sind nur bei Schaltflächeninstanzen zulässig	Ein Handler für Schaltflächenereignisse wurde in einer Bildaktionsliste oder in einer Aktionsliste einer Filmsequenzinstanz deklariert. Schaltflächenereignisse sind nur in Aktionslisten von Schaltflächeninstanzen zulässig.
Sequenzereignisse sind nur bei Filmsequenzinstanzen zulässig	Ein Handler für Sequenzereignisse wurde in einer Bildaktionsliste oder in einer Aktionsliste einer Schaltflächeninstanz deklariert. Sequenzereignisse sind nur in den Aktionslisten von Filmsequenzinstanzen zulässig.

INDEX

A

Absoluter Zielpfad 119
ActionScript
- Flash 4 im Vergleich zu Flash 5 18
- JavaScript-Unterstützung 18
- neue Funktionen 17
- Optimierung 20
- Vergleich mit JavaScript 17

Aktion
- trace 172

Aktionen
- Bildaktionen 48
- Festlegen von Filmsequenzen als Ziel 126
- kontextsensitive Hilfe 21
- neue Funktionen 18
- Schaltflächenparameter 47, 49
- Vergleich mit Methoden 127
- zum Steuern von Filmen zuordnen 128

AllowScale FSCommand 158
Angepasste Oberfläche 133
- erstellen 138
- xch (Filmsequenz) 139

Anhängen von Filmsequenzen 132
attachMovie (Methode) 126
attachMovieClip (Methode) 132
- Argumente 132

Ausgabefenster
- Objekte auflisten (Befehl) 171
- Variablen auflisten (Befehl) 171
- verwenden 170

B

Bedienfeld „Sequenzparameter“
- ersetzen durch angepasste Oberfläche 138

Beziehungen zwischen übergeordneten und untergeordneten Elementen 114
Bildaktionen
- Erstellen 48
- Zuweisen zu Schlüsselbildern 48

C

Core JavaScript Guide 18

D

Debugger
- aktivieren 164
- Anzeigeliste 166
- Filmeigenschaften 169
- Flash Debug Player 163
- im Webbrowser aktivieren 165
- Kennwort 164
- Statusleiste 165
- Überwachungsliste 167
- Variablen 166
- verwenden 163

Dialogfeld „Eigenschaften Symbolverknüpfung“ 132
Dialogfelder in Formularen 154
Drag Movie Clip (Aktion) 131
droptarget (Eigenschaft) 131
duplicateMovieClip (Aktion) 117
Duplizieren von Filmsequenzen 132

E

Ebenen
- absoluter Pfad 120
- Angabe im Zielpfad 120
- Hierarchie 113
- laden 129
- Laden von Filmen in 112

ECMA-262-Spezifikation 17
- tellTarget (Aktion) 127

Einfügen von Zielpfaden 123
Entfernen
- Filmsequenzen 132
- geladene Filme 129

Erstellen
- Smart-Filmsequenzen 133

Exec FSCommand 158

F

Fehlerbehandlung
  Objekte auflisten (Befehl) 171
  Prüfliste 163
  Überblick 161
  Variablen auflisten (Befehl) 171
  Verwenden der Aktion trace 172
  Verwenden des Ausgabefensters 170
Filme
  entladen 129
  ersetzen durch einen geladenen Film 129
  Laden von zusätzlichen 129
Film-Explorer
  anzeigen 119
Filmsequenzen
  anhängen 132
  Anzeigen der Hierarchie 113
  Austausch 139
  Definieren von Sequenzparametern 134
  duplizieren 132
  entfernen 132
  gemeinsam nutzen 132
  hierarchische Beziehungen 114
  Info 111
  steuern 123
  ziehen 131
Flash Debug Player 164
Flash Hilfe
  Aktionen 21
Formulare
  erweiterte Interaktivität 153
  Überprüfen von Daten 156
FullScreen FSCommand 158

G

Geladene Filme
  steuern 123
getBounds (Methode) 126
getBytesLoaded (Methode) 126
getBytesTotal (Methode) 126
globalToLocal (Methode) 126

H

Hierarchie
  Filmsequenz 113
  Filmsequenzen mit untergeordneten Sequenzen 114
hitTest (Methode)
  Steuern von Filmen 126

I

ISO-8859-1-Zeichensatz 18

J

JavaScript
  Developer Central 18
  internationaler Standard 17
  unterstützte Sprache 18
  Vergleich mit ActionScript 17
  with (Anweisung) 118

K

Kennwort
  Debugger 164
Kommunikation
  zwischen Zeitleisten 116

L

Liste der RGB-Farbwerte 417
loadMovie (Aktion)
  Ebenen 112
localToGlobal (Methode) 126

M

Methoden
  aufrufen 127
  Festlegen mehrerer Zeitleisten als Ziel 128
  Vergleich mit Aktionen 127
  zuordnen 128
MovieClip (Objekt)
  Steuern von Filmen 126

N

Netscape DevEdge Online 18

O

Oberflächenelemente
    angepasst 133
    Smart-Filmsequenzen 133
Objekte auflisten (Befehl) 171

P

_parent (Alias) 121
Punkt-Syntax
    Zielpfade 121

R

Relativer Zielpfad 119
removeMovieClip (Aktion) 132

S

Schaltfläche “Zielpfad einfügen” 123
Schlüsselbilder
    Zuweisen von Bildaktionen 48
Schrägstrich-Syntax
    Zielpfade 121
Sequenzparameter
    definieren 134
    setzen 136
    setzen für Smart-Filmsequenzen 137
    zuweisen 133
Shift-JIS-Zeichensatz 18
ShowMenu FSCommand 158
Smart-Filmsequenzen
    erstellen 133
    Setzen von Sequenzparametern 137
Steuern von Filmen
    Anforderungen 123
Steuern von Filmsequenzen
    Methoden 126
swapDepths (Methode) 126

T

targetPath (Funktion) 125
Testen von Bildaktionen 49
this
    Alias für aktuelle Zeitleiste 121

U

Überprüfen der eingegebenen Daten 156
Überwachungsliste
    Debugger 167
unloadMovie (Aktion) 129

V

Variablen
    im Debugger ändern 166
    übergeben mit Smart-Filmsequenzen 133
    überprüfen 156
Variablen auflisten (Befehl) 171
Vereinigung europäischer
        Computerhersteller (ECMA) 17
Verknüpfen von Filmsequenzen 132

W

with (Aktion) 118
    Festlegen mehrerer Zeitleisten als Ziel 128
Wrapperaktion 20

X

xch (Instanzname) 139

Z

Zeitleisten
    _parent (Alias) 121
    Festlegen eines Ziels für mehrere Aktionen 128
    Kommunikation zwischen 116
    mehrere 112
    steuern 126
Ziehen von Filmsequenzen
    auswerten 131
Zielauswahl
    duplicateMovieClip (Aktion) 117
Zielpfade 119
    angeben 123
    Ausdruck 125
    Ebenennamen 120